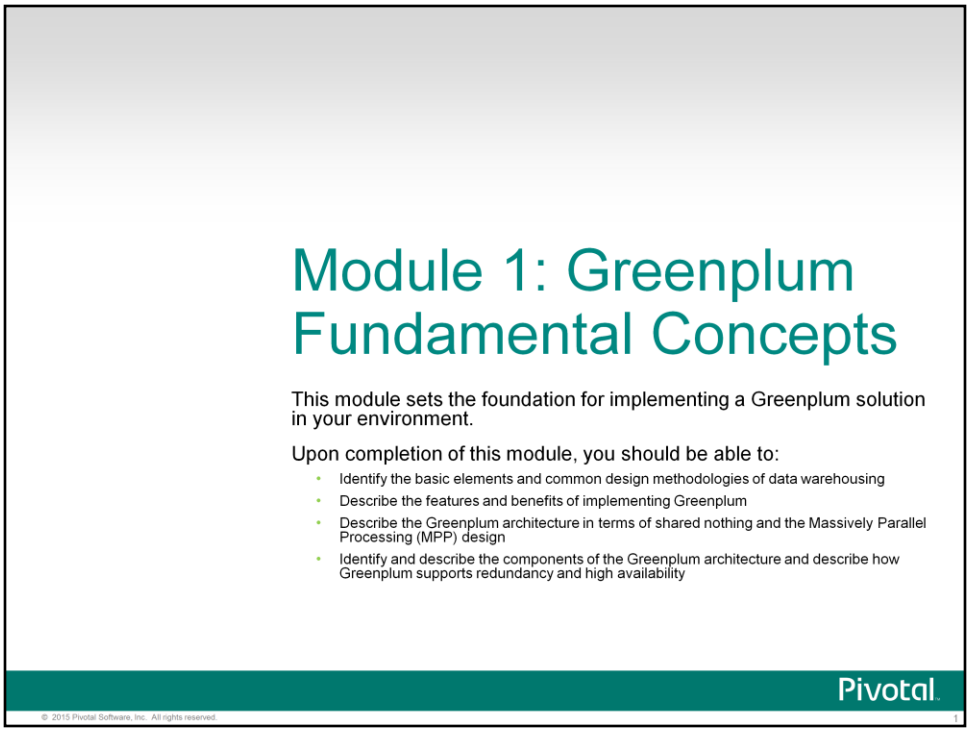
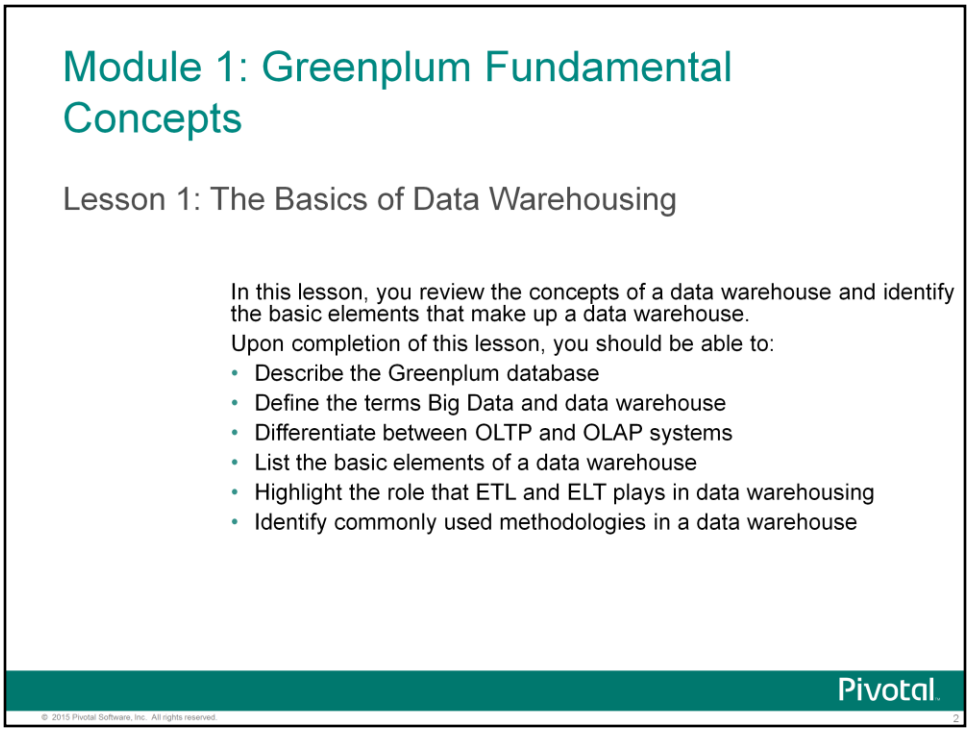
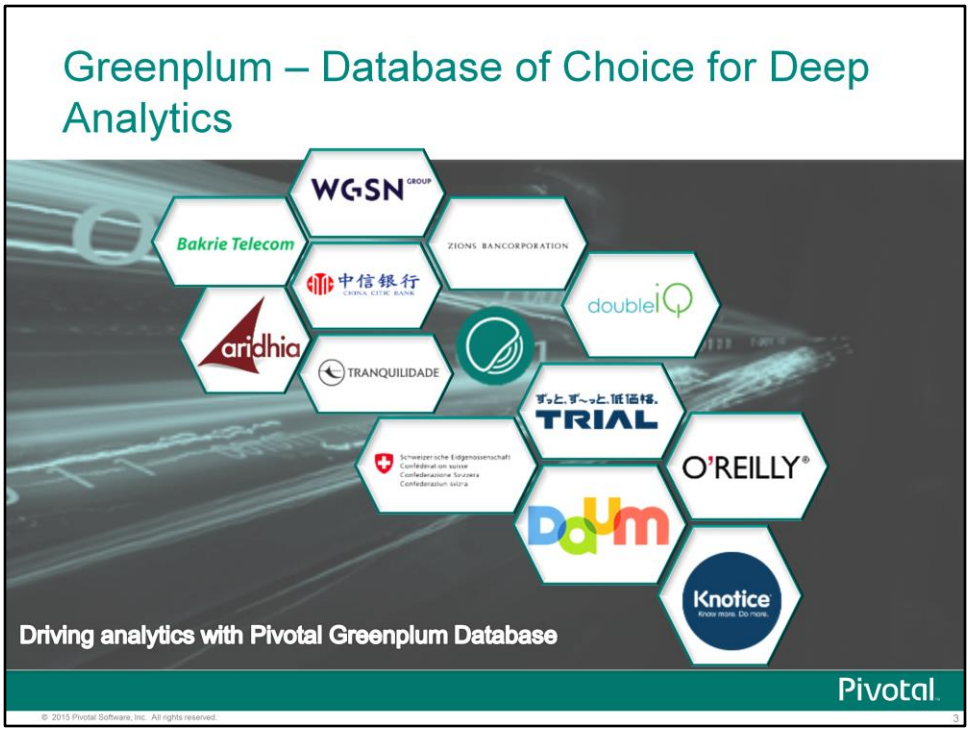


Greenplum is a solution built to support the next generation of data warehousing and large-scale analytics processing. In this module, you will learn foundation concepts that will be applied when implementing the Greenplum solution both during this course and in your environment. You will:

- Examine the basic elements of a data warehouse and identify common design methodologies used when implementing a data warehouse.
- List and describe the features available from Greenplum and the benefits of implementing Greenplum in an environment.
- Examine a high-level overview of the shared-nothing massively parallel processing (MPP) design.
- Identify and describe the components of the Greenplum architecture and describe how Greenplum supports redundancy and high availability.



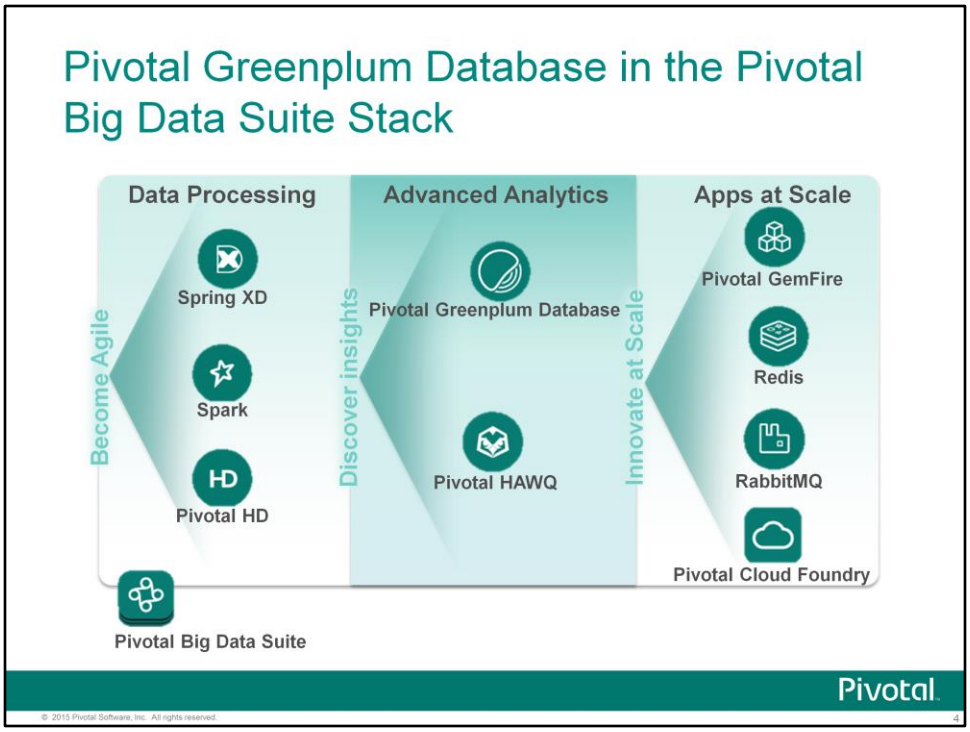
The lesson introduces you to the concept of data warehousing as well as the basic elements and methodologies used in building and implementing a data warehouse. You examine concepts meant to differentiate between online transaction processing systems and online analytic processing systems. Additionally, you examine the role that extract, transform, and load (ETL) and extract, load, and transform (ELT) play in creating your data warehouse.



Greenplum is the database of choice for deep analytics. Pivotal Greenplum Database, designed for business intelligence and analytical processing, adds value in that you can now ask questions you were never able to ask before and in return, obtain valid and solid answers.

Greenplum Database is built to support the next generation of Big Data warehousing and large-scale analytics processing. It stores and analyzes terabytes to petabytes of data. The Greenplum Database was conceived, designed, and engineered to allow customers to take full advantage of large clusters of increasingly powerful servers, storage, and internet switches.

The number of companies looking for a solution to tackle the growth and management of data in their environment is ever increasing.



Greenplum Database has evolved to become a backbone product for advanced analytics of the Pivotal Big Data Suite. The core principles behind the Big Data Suite allow an organization to:

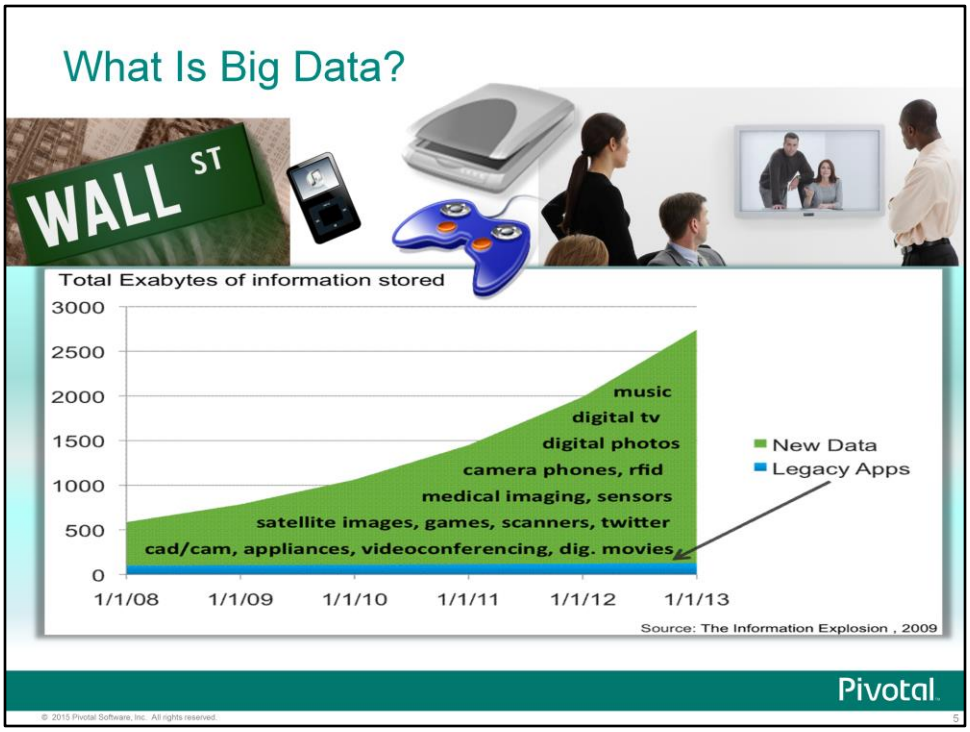
- Become agile where they are able to more quickly and manage assets in the business data lake. The ability to take disparate sources of data and provide a centralized access point to that data is becoming a necessity in today's move towards growth and accessibility.
- Discover more insights by employing predictive analysis against large amounts of data. The combination of an enterprise ad-hoc solution and analytical data warehouses offer flexibility for gaining insight into large datasets that would otherwise be difficult to parse.
- Innovate at scale by acting on data to create customized experiences for different customers, including deploying mobile applications from a single framework to multiple sources with context-aware features based on real-time, in-memory data stores and reliable message queues.

With the availability and consolidation of all these toolsets, Greenplum Database, introduced with cloud storage as part of the Big Data solution has now moved to provide strong support in the enterprise data lake.

Greenplum can be physical or virtual, can be executed on any type of hardware, and can provide the flexibility customers are looking for.

Scalability, maintainability, and real-time provisioning in a virtual environment has great appeal as the industry moves towards enterprise data cloud.

Before continuing to discuss the Greenplum database, let us fully explore the concept of Big Data and the data warehouse.

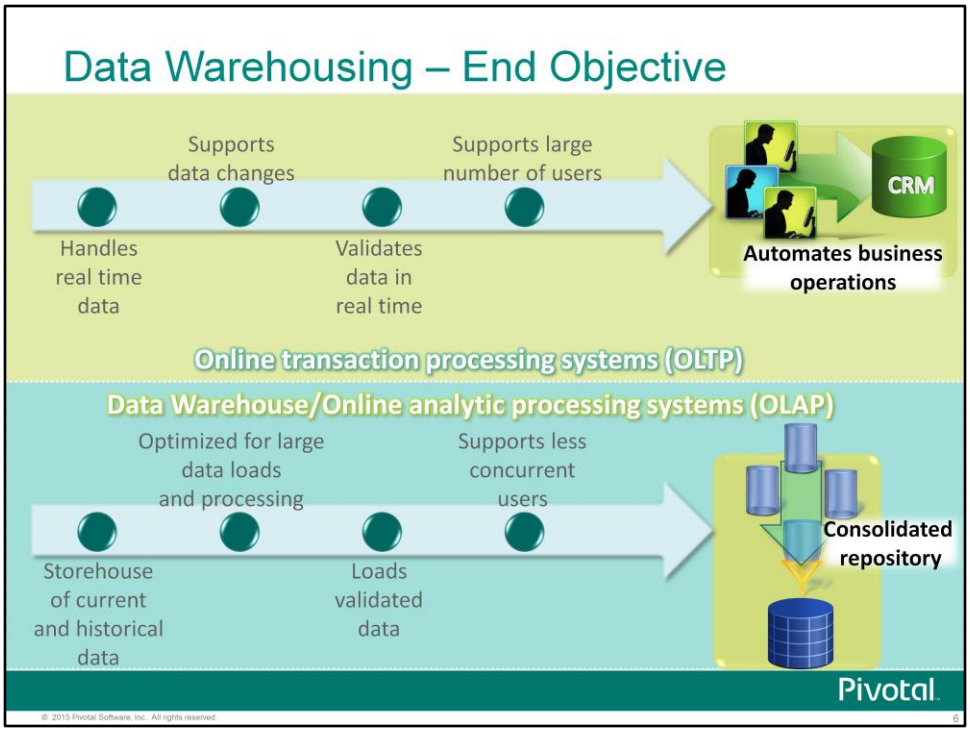


Big Data refers to the tools, processes, and procedures used to create, manipulate, and manage very large data sets, on the order of terabytes and petabytes of data.

Big data is changing and Greenplum continues to lead in changing how it is perceived. In yesterday's data warehouse and analytic infrastructure, big data is proprietary, expensive, monolithic, loaded once a, and ultimately, slow. The demands of data and business are changing.

Time to business value is now measured in minutes. Data may need to be loaded every 60 seconds and support must be there to analyze data in in smaller time frames. This helps a company spot trends that are occurring in real-time. This change represents the move to the next generation of Big Data, where queries can be performed in real time against petabytes of data.

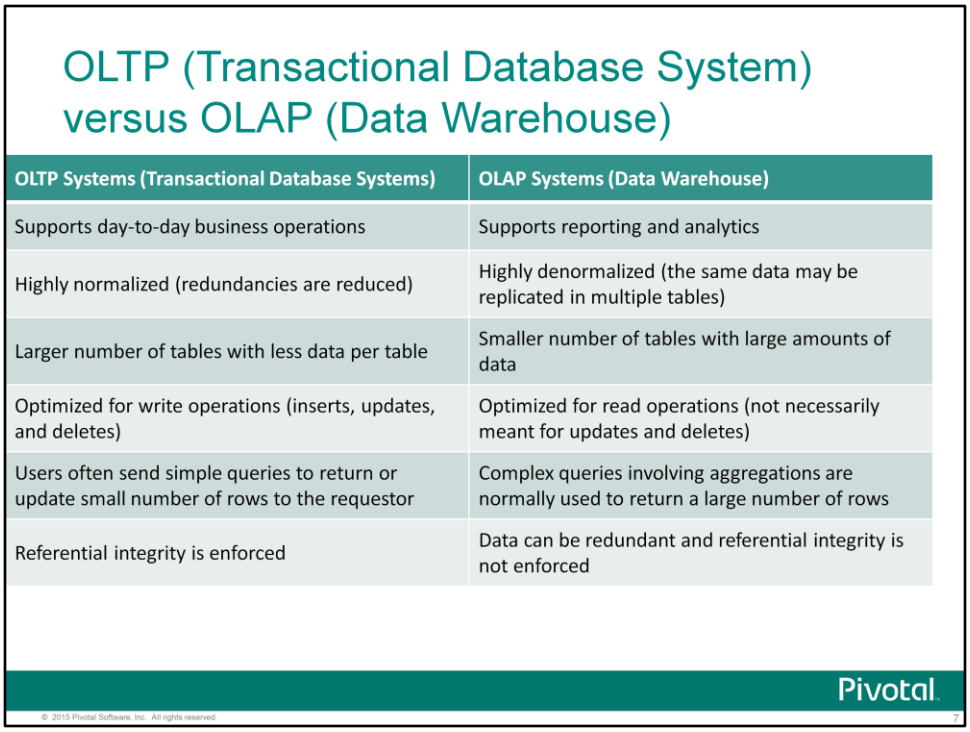
The proliferation of data presents a challenge in traditional databases, data warehousing, business intelligence, and analytics. Real-time processing represents the merging of OLTP and DW. The data warehouse has to be more agile to support real-time analysis.



To understand the end-goal of a data warehouse, it is important to distinguish data warehouses from transactional processing systems.

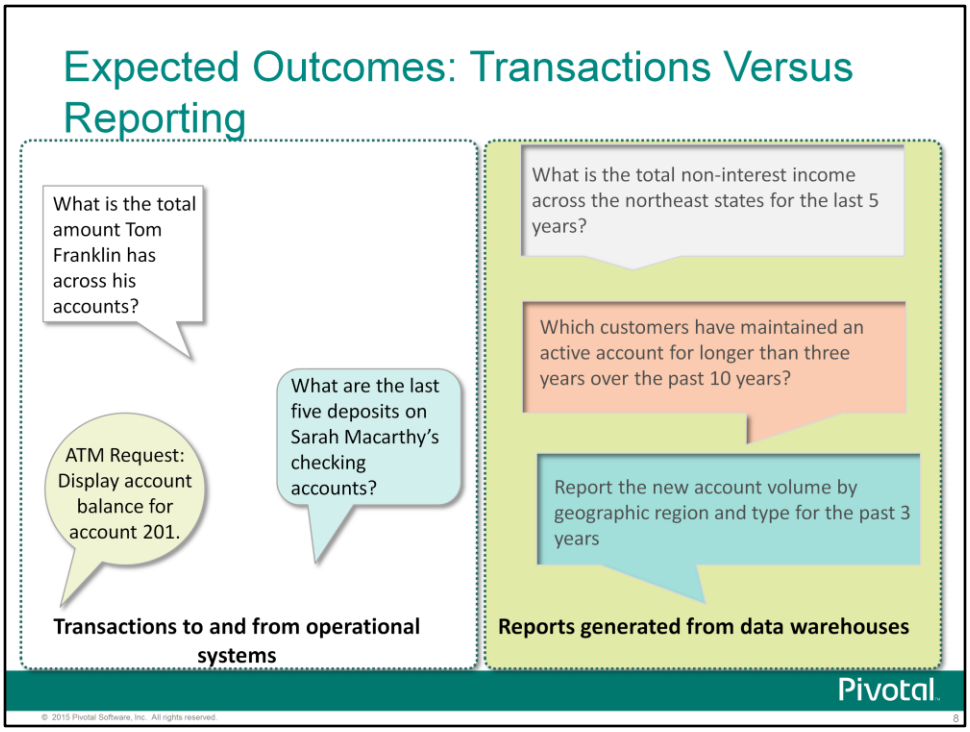
Online transaction processing systems, or OLTP systems, automate day-to-day business operations and processes by handling data in real time and allowing modification of existing data, removal of data, and addition of new data. All data is validated in real time. These systems normally support thousands of concurrent users, handling very fast and simple ad-hoc queries.

A data warehouse, also referred to as an online analytic processing system, is a consolidated repository designed for reporting and analysis. Storing current and historical data, it is optimized for bulk loads and processing of large amounts of data. These systems often rely on bulk loads to load current data into the system. Years of data may be maintained within the database, allowing long term analysis of data. Lower number of concurrent users perform more complex queries that may take minutes or hours to complete.



Businesses use a combination of these complementing technologies to support business processes and analytic processing. On-line transaction processing systems are used to support day-to-day business processes such as would be found in sales transactions and invoicing systems, ATM or other daily banking transactions, or tracking for logistics and shipping systems, just to name a few. The data in OLTP systems is highly normalized, reducing redundancies, increasing the number of tables, and normally creating a logical grouping of tables. This data normalization improves data integrity and reduces the chance of anomalies. This type of environment is optimized for write operations where you experience frequent data changes affected by inserts, updates, and deletes. OLTP systems are also well tuned for common read queries that return a low number of data records.

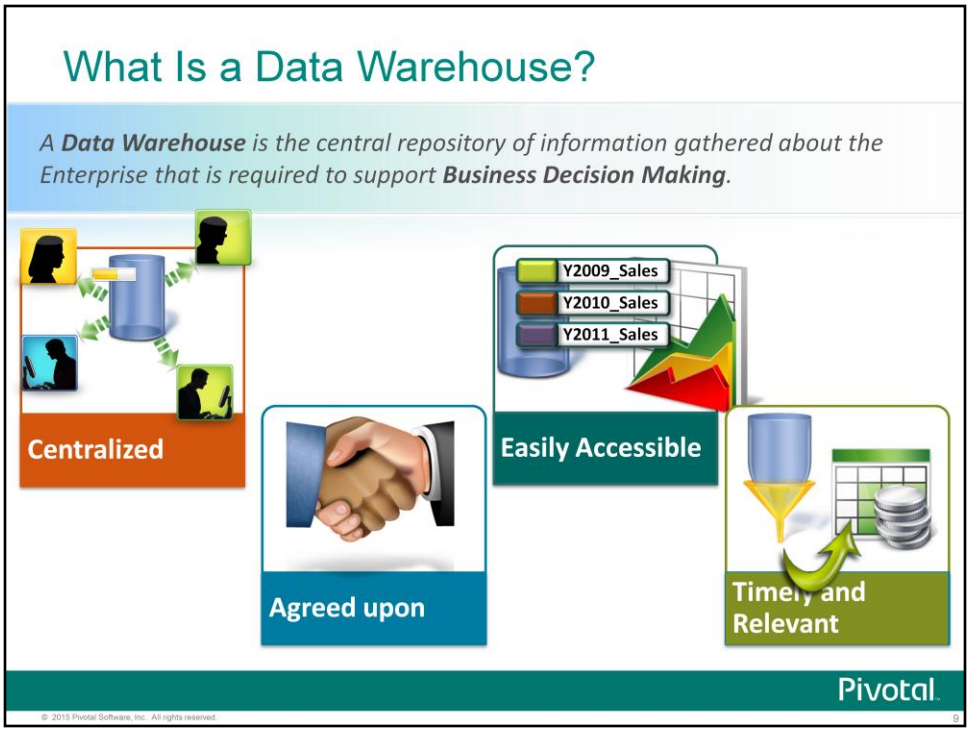
Online analytic processing systems are designed to handle complex queries that may involve multiple aggregations. Data in these systems is denormalized with fewer tables, making it more attractive for read operations. Data may be refreshed but is normally not updated. The focus of this data is to provide line of business users and executives the information they need to help drive business decisions by looking at historical and temporal data.



Transactional or operational systems deal with day-to-day queries that operate the business. Users of operational systems initiate transactions, take orders, invoice customers, ship orders, or book events. What they ask of, or add to transactional systems are usually small (on the order of a single transaction at a time) and require fast response times.

Data warehouses are historical tracking systems that allow you to keep a bevy of information over the course of years instead of on a daily basis. In a banking scenario, it may be keeping track of the daily interest rate on an account over a period of years, whereas a transactional system keeps track of the current interest rate. This allows a consumer of this system to track trends over large periods of time. It also means that data in this infrastructure is not only redundant, in that it is repeated in different tables, but you may find multiple iterations of data as it is applied to a single entity.

Continuing with the interest rate example, the transactional system will show the current interest rate as it applies to an individual's account. In a data warehouse, that same customer may show up 100 times in the table because the interest rate on their account has changed over the past 3 years. The ability to track this type of information may help identify trends over the long term in customers' banking behaviors.



A data warehouse is a culmination of information gathered about the enterprise. The information, which can consist of a variety of topics including sales data, organizational data, operational data, and inventory data, is used to support business decision making.

The data warehouse acts as a single source of truth, often times collecting data from multiple stores, making it easier to retrieve and analyze the data from a centralized collection.

In a data warehouse, data must be:

- **Centralized** – Data, while it may be accessible from multiple locations or other data marts, should be centralized when in a data warehouse
- **Agreed upon** – If the data is to be a single source of truth, it must be agreed upon. Data must be clearly defined so that it is useful when analyzed. This "agreed upon" terminology refers to the metadata that will be used across all business functions.
- **Easily accessible** – Data should be clearly labeled and consistent. Inconsistent data yields inconsistent reports which can ultimately affect business decisions.
- **Timely and relevant** – Data can give a company the competitive edge, if it is readily available and relevant to business needs. You should therefore be able to extract data as quickly as possible to help a business not only meet its goals, but also to gain a competitive edge in their market.

Data Warehouse – the Centralized Repository

*The Data Warehouse must contain a **single view** of the business, or the **Single Version of the Truth**, in order to be valuable to the enterprise.*

- Conflicting answers are possible *if* a company has multiple databases or data marts
- A centralized data warehouse should be the platform to unify all the versions of "the truth", coming from the company's transactional systems
- Lack of a well-defined "Single Version of the Truth" can lead to a distrust of information

Pivotal

A centralized repository presents a single view of the data. If you do not have a centralized repository, this creates problems for aggregating and analyzing your data, including creating conflicting answers to business questions.

This centralized repository should act as the single version of truth within the enterprise. This eliminates the need to validate the data by accessing other databases or data marts within the enterprise. The organization must be able to trust that the information contained within the data warehouse is valid and therefore, the single version of truth for the enterprise.

Data Warehouse – the Data Must Be Agreed Upon

*The components of the Data Warehouse, subject areas, and the data elements should be **clearly defined and agreed upon** by the Business!*

Agreed upon

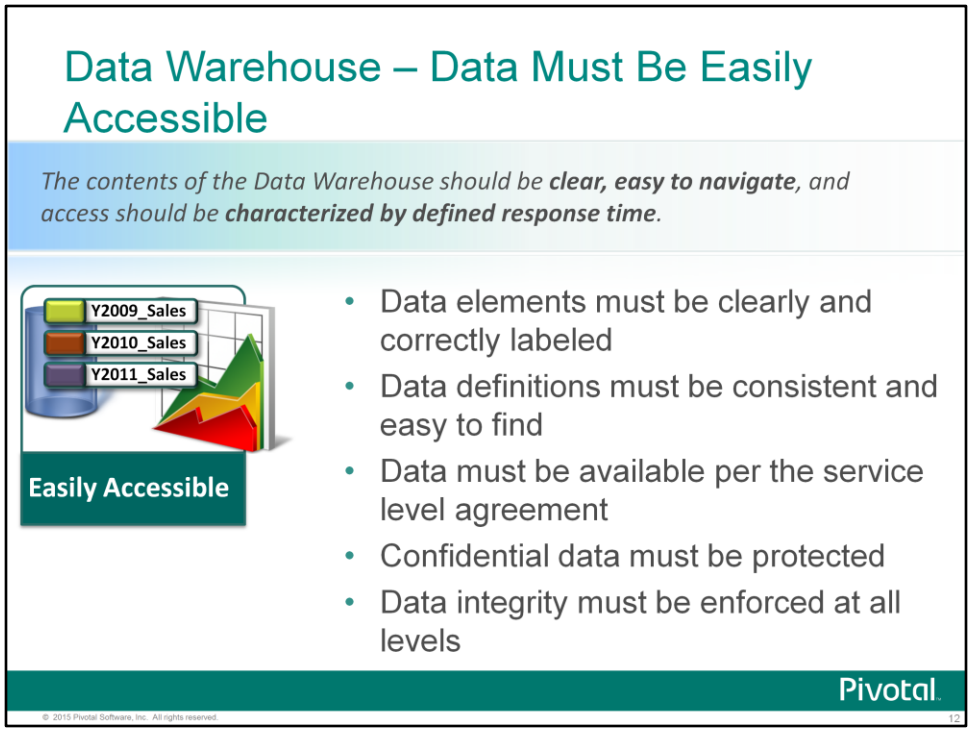
- Report metrics should have common and consistent definitions and business logic
- Achieving agreement among stakeholders that provide and extract data from the Data Warehouse is very important and often difficult task for Cross Functional Analysts

Pivotal

Data that is not clearly defined and agreed upon may result in inaccurate or misleading reports.

Consider the following scenario: A sales report indicates the number of servers sold by field sales in the last quarter. For this report, sales figures include initiated sales contracts, or when the order has been placed. The sales organization within the company uses this business logic to determine their sales figures. Finance however relies on a paid invoice to determine that a sale has been. A single report relying on the first business logic yields skewed sales figures for organizations that determine quarterly sales figures based on the second logic.

Data must therefore be agreed upon to be useful to the entire business as a whole. This requires clear definition in business logic and rules.



Collecting and analyzing data relies on data that is:

- Clearly defined and correctly labeled
- Consistent and easy to find
- Available at the defined performance level
- Confidential
- Valid, where data integrity is maintained at all levels

Invalid, incorrect, or mislabeled data makes collection and analysis a difficult task. To extract data, an analyst must know precisely what data to look for and expects that the data is valid at the time of collection. To protect privacy, confidential data must be protected from unauthorized access.

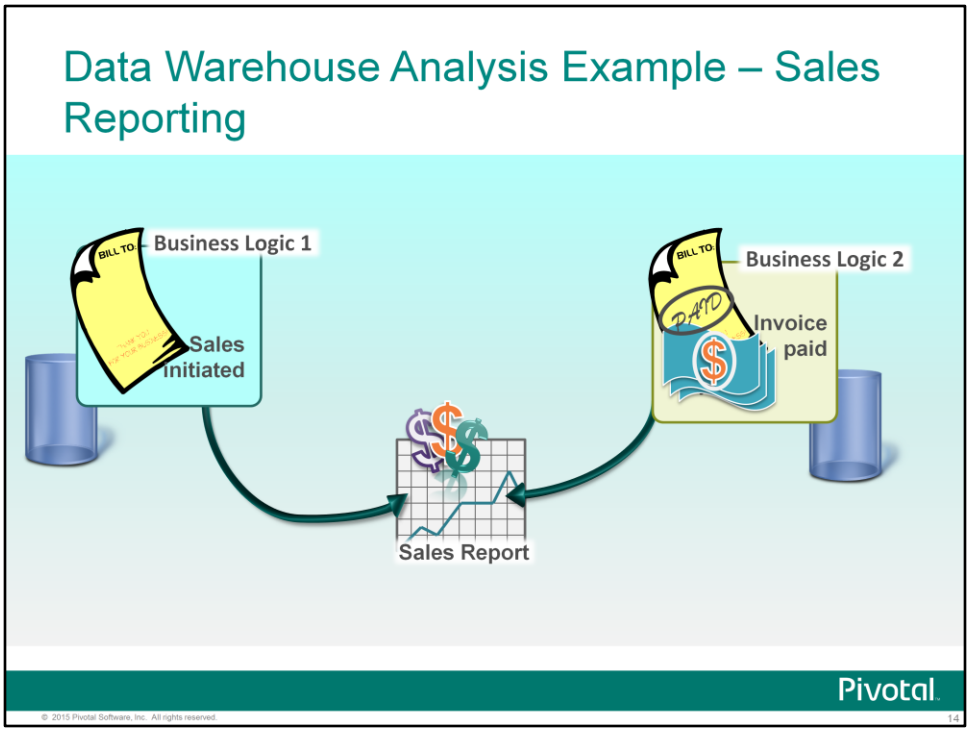
Data Warehouse – Data Must Be Timely and Relevant

*Data Warehouse must be refreshed as often as required for **relevant reporting and analysis** to **support timely decision making**!*

- The data warehouse must have the *right* data to support decision making
- Stale data translates to late or poor decisions
- The only true output from a data warehouse consists of the business decisions made after reviewing the information retrieved from the data warehouse

Pivotal

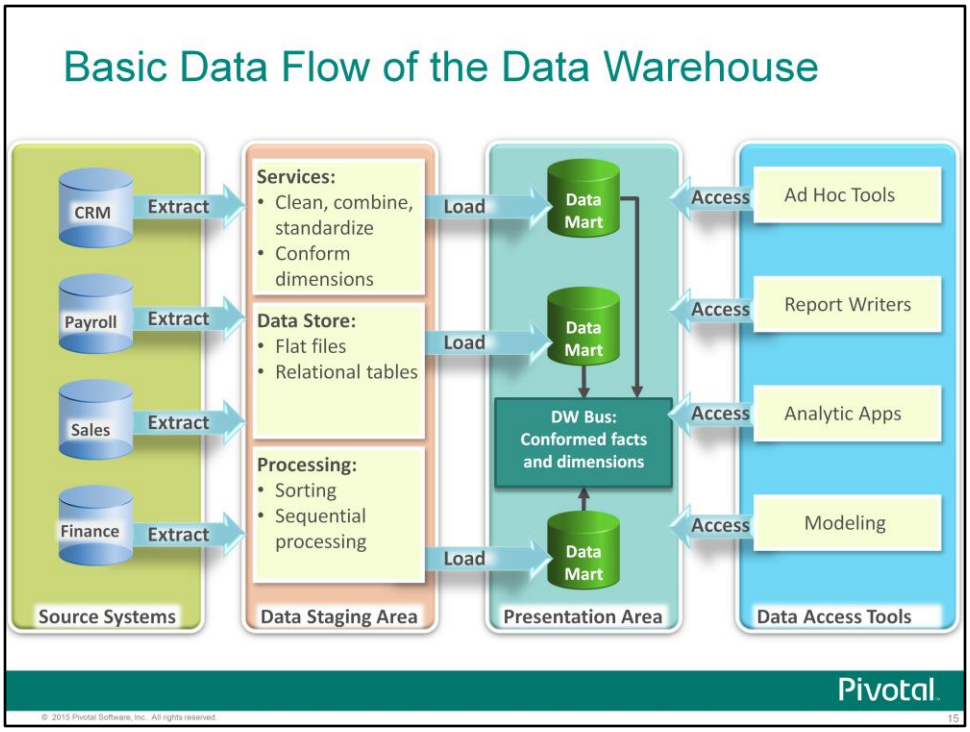
A financial institution needs to analyze trading volumes on the NASDAQ. For this information to have any relevance, freshly acquired data must be available every five minutes. The ability to quickly analyze new data allows an organization to make better business decisions.



Here is a simple case where there are two business rules being applied to data to generate a single report for an organization or company. One system looks at sales from the point where sale orders were generated. A second system looks at sales from the point at which the sales order has been completed and the invoice has been paid. The second system will be more tightly coupled with inventory and billings than the second system. Both systems use the term sales to indicate the end result of the business logic applied.

However, a report generated takes the term, or field, sales, and generates a report to show how the organization as a whole is doing. Because the two systems apply different logic to the sales terminology, the report does not necessarily reflect what the organization is looking for.

By applying basic data warehousing concepts that examines the business models, actors, business logic, and other requirements, the potential for inaccurate reporting and analysis can be greatly reduced.

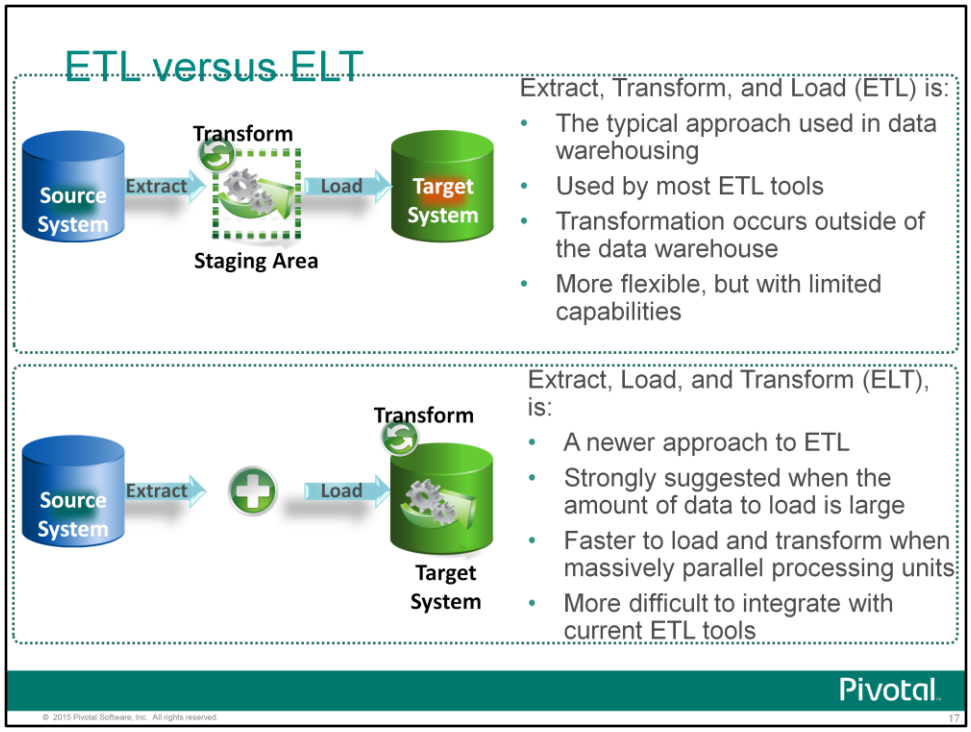


The basic components or elements of a data warehouse include:

- **Source System** – An operational system of records which function is to capture the transactions of the business. There is usually more than one.
- **Data Staging Area** – A storage area and set of processes that cleanses, combines, removes duplication and prepares the data for insertion into the Data Warehouse. The staging area may be spread over a number of systems. It does not provide query and presentation services to the user.
- **Operational Data Store (ODS)** – A subject-oriented, integrated, current and detailed-only collection of data in support of an organization's need for up-to-the-second operational reporting. Could be used as a staging area for Data Warehouse.
- **Data Mart** – In the presentation area, you have the Data Mart, a dependent, or independent area of data storage, usually related to one subject area. An Independent Data Mart stands on its own and bears no relation to any other subject area in the Data Warehouse. A Dependent Data Mart has relationships defined outside of its own subject area.
- **Business Intelligence** – A set of tools and business logic in the data access tools area that turns the data from the Data Warehouse into useful information for Decision Support.

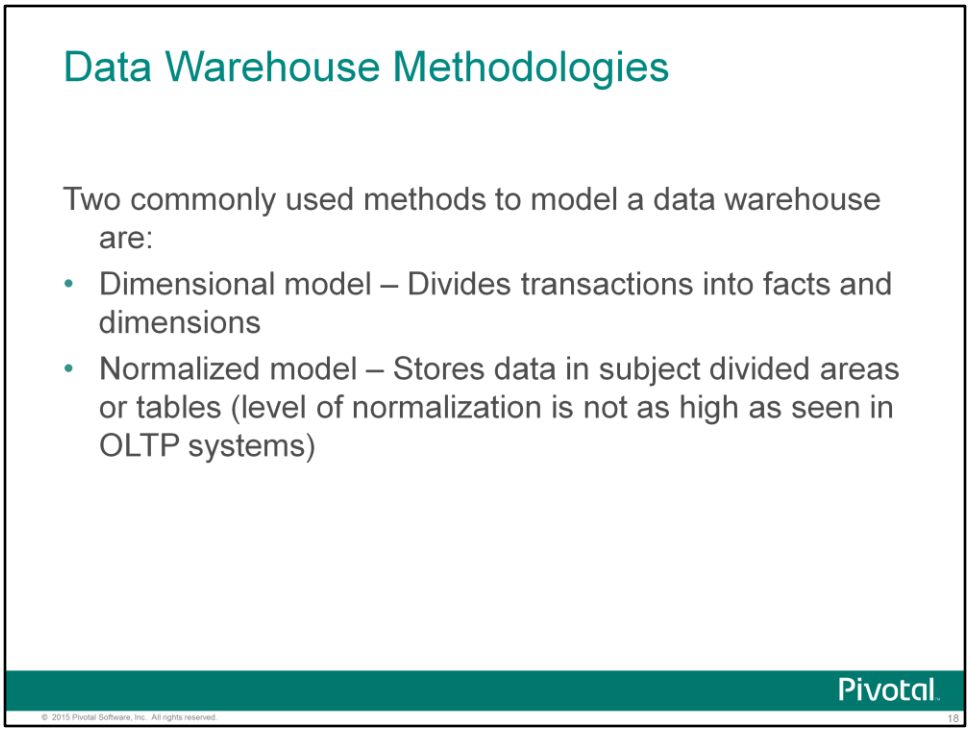
Basic Data Flow of the Data Warehouse (Continued)

- **Metadata** – All of the information in the Data Warehouse Environment that is not the data itself. Often called the "Data about the Data".
- ETL – a set of processes in the Data Warehouse system to:
    - Extract the data from external systems
    - Transform it according to the business logic
    - Load it into to target DW tables



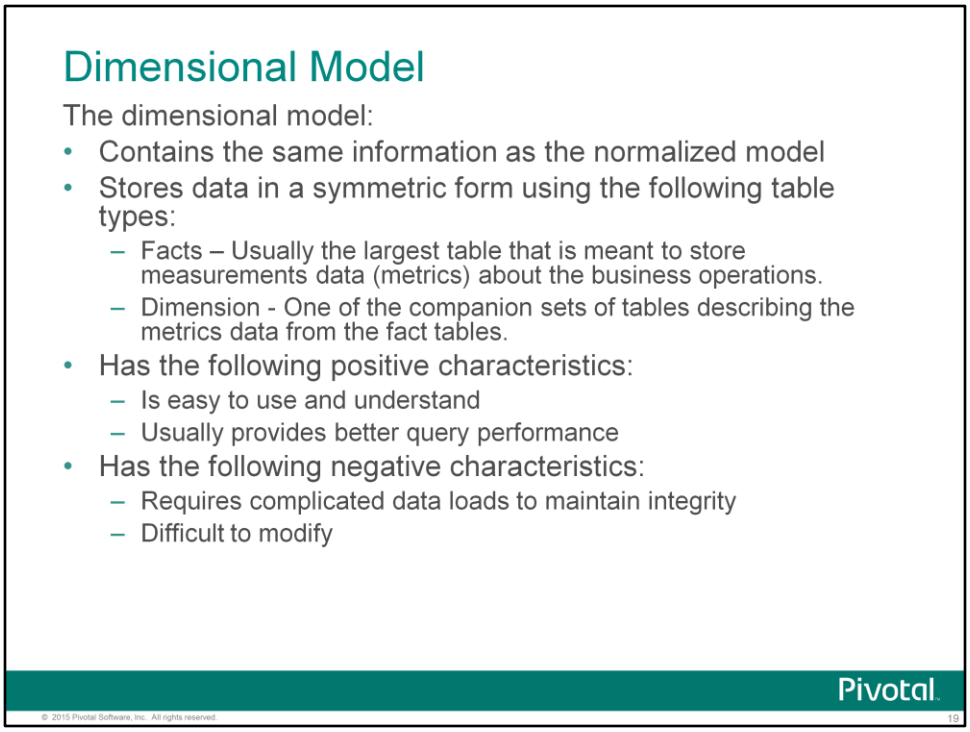
In ETL, or extract, transform, and load, you extract the data from a source system, perform the transformation of the data on the ETL or staging system, and load the data to the target or presentation area, which would be the data warehouse. While the solution is flexible, it does have several limitations. One major factor to consider when designing a solution using ETL is that the solution is tied to hardware to perform the transformation. The source data is moved from one or more source systems or data marts and must be cleaned, verified, and transformed based on the data format of the output, before moving to the warehouse. The industry has multiple tools that support the use of ELT in the data warehousing environment.

In ELT, or extract, load, and transform, you extract the data from the source system and perform the load and transform in the target system or presentation area. The staging area can be defined within the target system itself, allowing you to utilize existing hardware. This method is strongly advised when working with large amounts of data. While it is powerful, it is difficult to integrate with current tools such as Informatica or Data Stage. The toolset for this methodology is limited. However, this method allows you to take advantage of the powerful featureset offered by Greenplum, including the massively parallel processing units which allow you to perform the load and transformation much faster.



There are essentially two commonly used methods to model a Data Warehouse. This section will discuss each in detail. The two methods are:

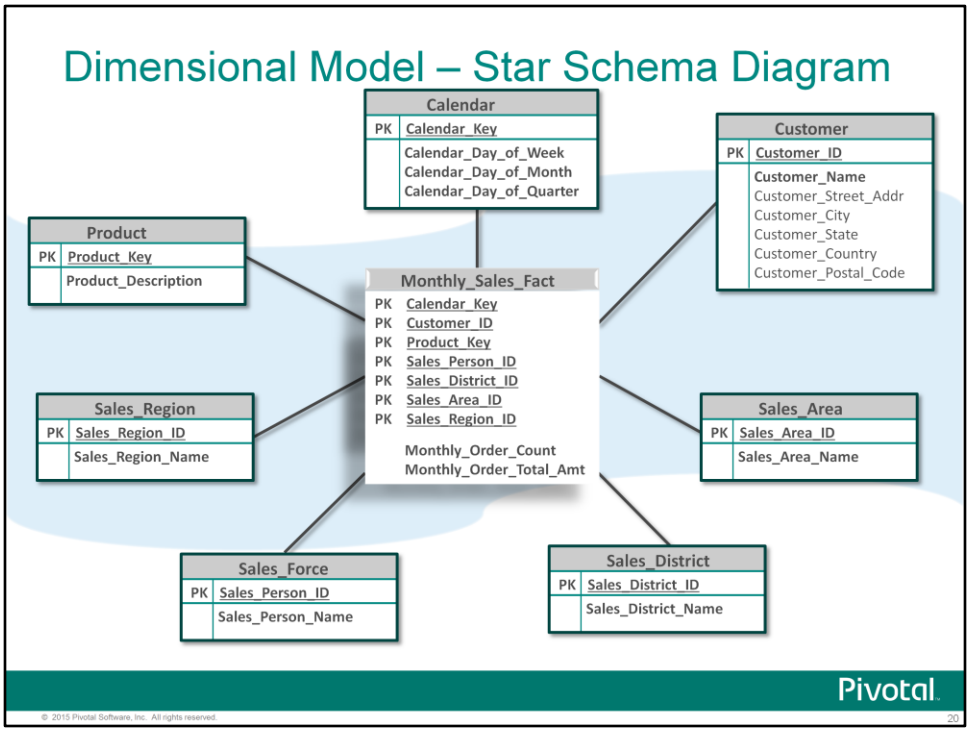
- **Dimensional model** – Divides transactional data into facts and dimensions. Facts usually stand for various measures and dimensions give a context information about the facts. This is the most common data warehouse model.
- **Normalized model** – Stores data in subject divided areas (tables) and derives its name from the Codd data normalization methods. The level of normalization for this approach is usually not as high as for the typical OLTP data models.



The dimensional model contains the same information that is supposed in the normalized model. It differs in that it stores data in a symmetric form using fact and dimension tables.

A fact table is usually the largest table and is meant to store measurement data, or metrics, about the business operations. The dimension table describes the metric data from the fact table. Each fact table row holds a foreign key ID that points to a row in a dimension table.

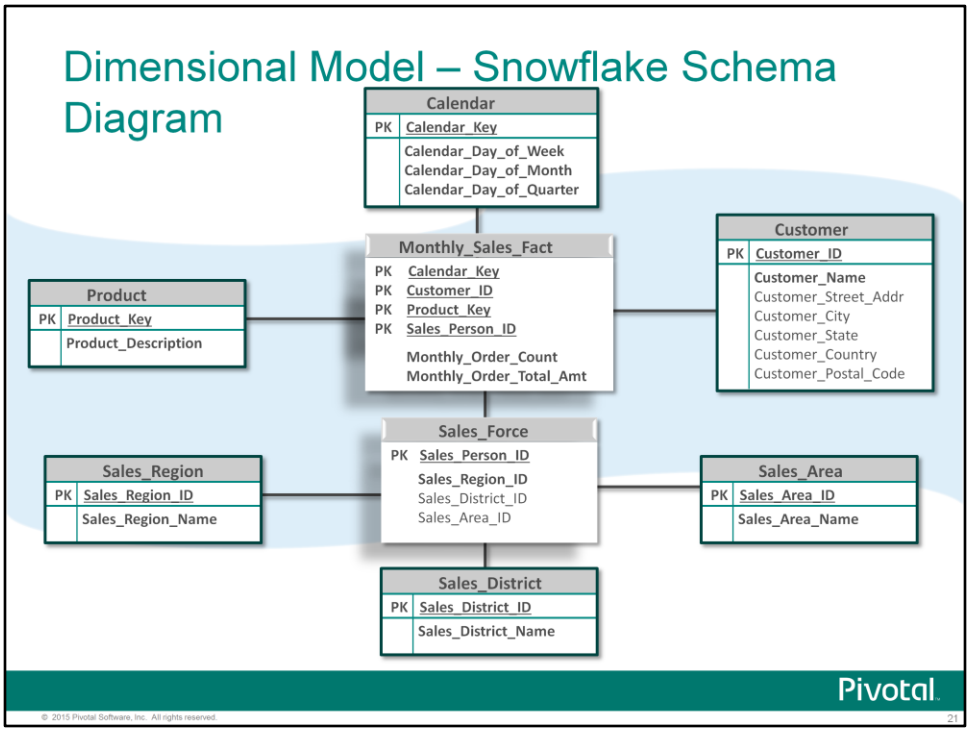
The dimension table is easier to understand and use and overall, performs better on query operations. That ease of use comes at a cost. It requires complicated data loads to maintain data integrity. As such, it can be difficult to modify once established. This makes it difficult to maintain the database in the face of business changes.



We will briefly examine two types of dimensional models that can be implemented: the star and snowflake schemas.

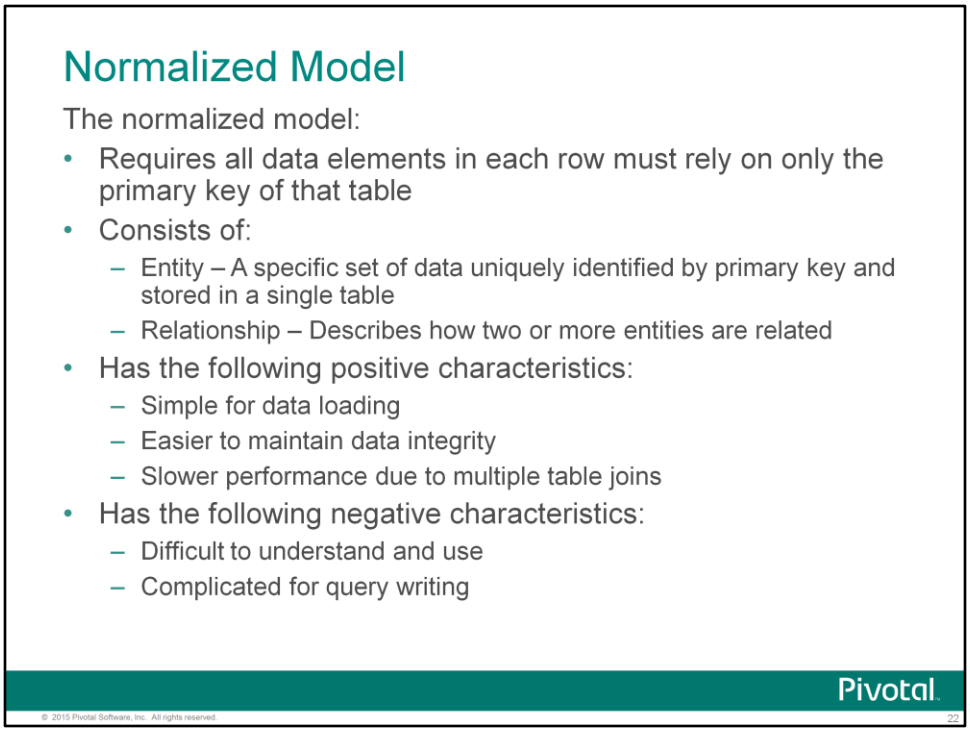
In the star schema, the fact table, `Monthly_Sales_Fact`, holds the main data. The remaining tables are dimension tables which describe the facts presented in the fact table. The fact table contains a compound primary key which are actually foreign keys from the dimension tables. The dimension tables are simpler, consolidating data in the most granular column, and are therefore rendered in second-normal form.

The fact table is usually in third normal form because all of the data depends on either one dimension or all of them, not a combination of just a few dimensions.



In the snowflake schema, multiple centralized fact tables connect to multiple dimension tables. The arrangement in the entity relationship diagram is such that the schema is represented in the form of a snowflake. With the snowflake schema, dimensions are normalized into multiple related tables. A single dimension table may therefore have multiple parent tables.

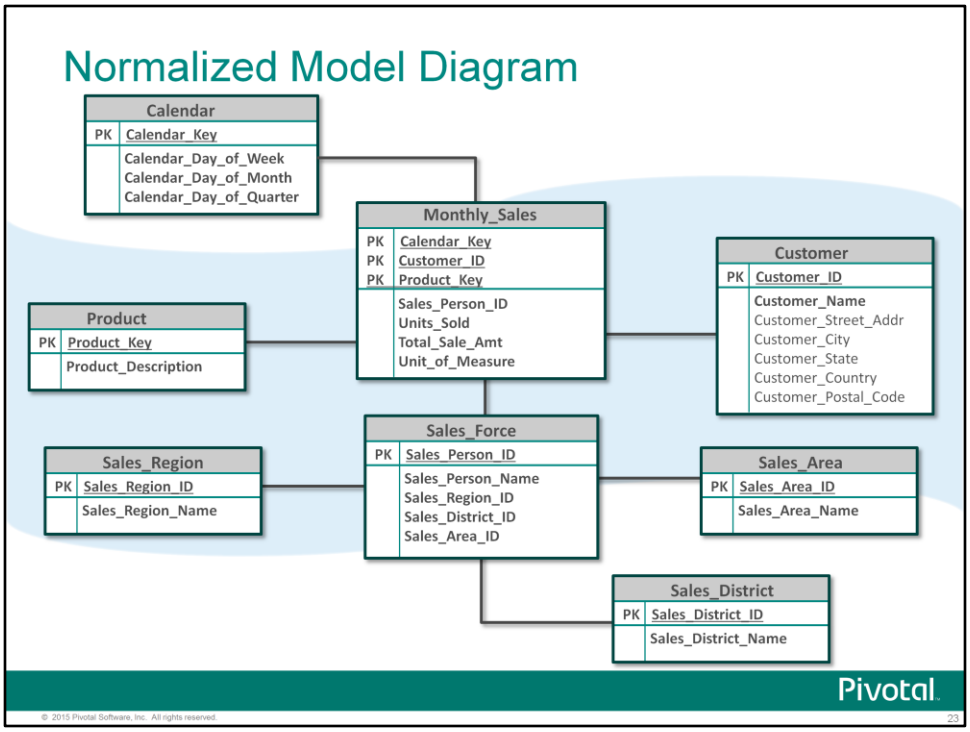
In this example, the fact table only has four primary keys, one for each table to which it is directly connected. A dimension table, `Sales_Force`, is directly associated with other dimension tables.



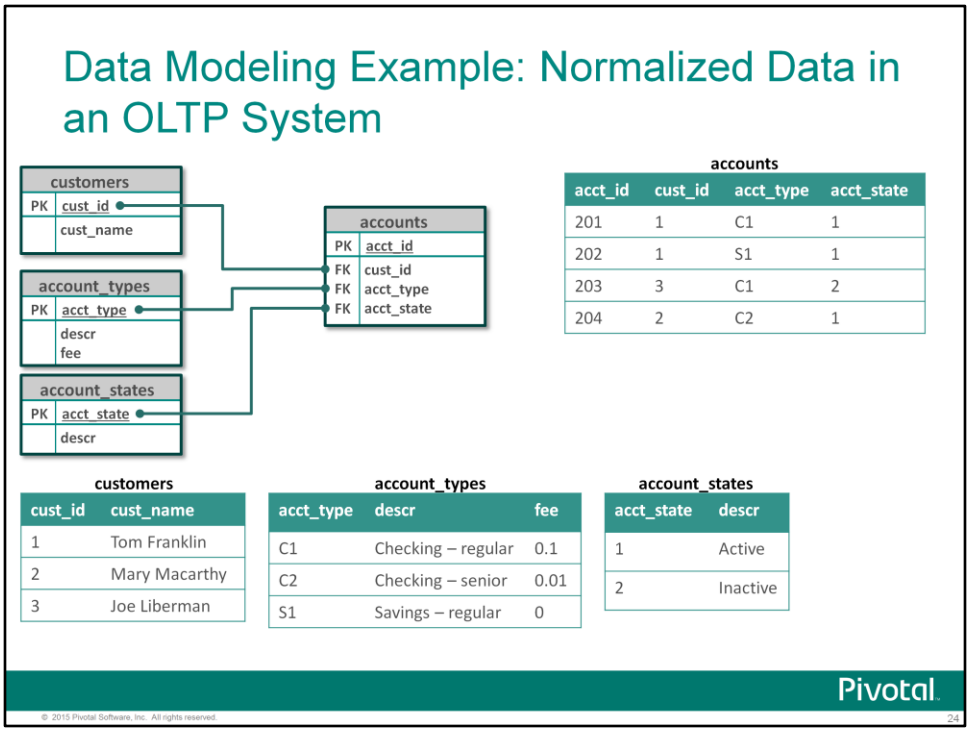
The normalized model, also known as the third normal form, is based on the normalization rules defined by E. F. Codd. It maintains that all of the data elements in each row must rely on the primary key and only on the primary key of that table. Tables are grouped together by *subject areas* that reflect general data categories.

The main components of the normalized model are the entity and relationship. The entity is a specific set of data uniquely identified by the primary key and stored in a single table. The relationship describes how two or more entities are related to each other. For example, it may describe how a customer is related to a product.

The normalized model is simple for data loading and easier to maintain for data integrity. On the negative side however, it is difficult to use and understand. It can be complicated when developing queries, and can be hampered by performance issues due to the need to join multiple tables together to gather the information requested by the query.

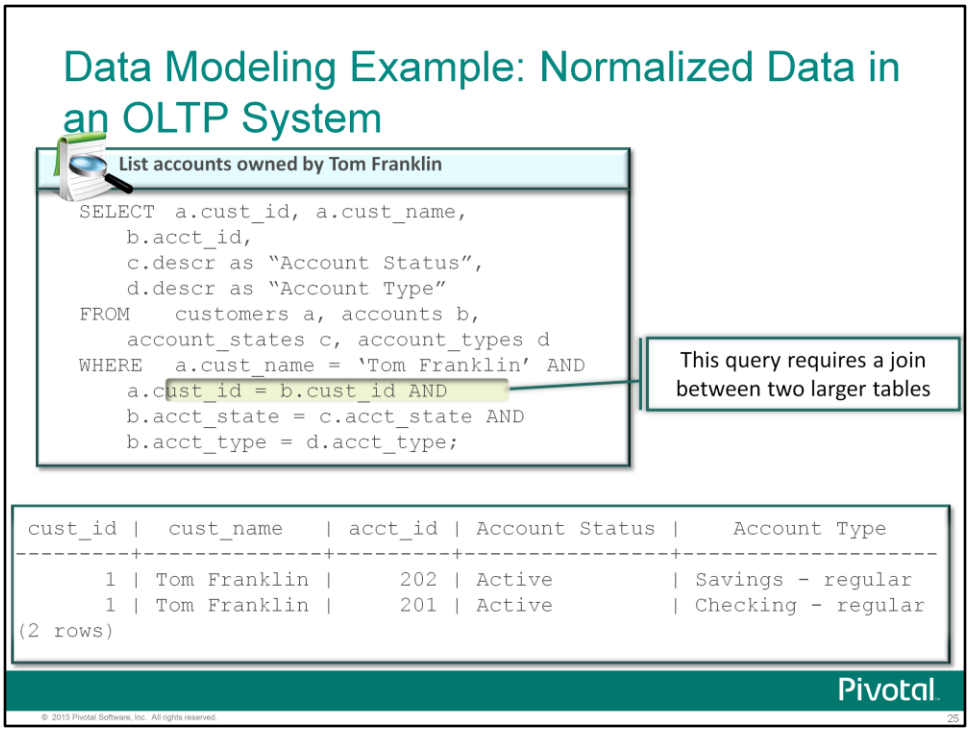


The diagram reflects the normalized model as you would find in an OLTP system, not in a data warehouse environment. Each entity is defined within its own unique table. Though the sales table has a compound primary key that consists of multiple foreign keys, sales is itself an entity and is described by those primary keys defined. It is easier to extend this diagram if you need to add additional entities – you would define separate table. However, you can see how quickly performance may decrease when you require multiple tables to be joined together to respond to a query.

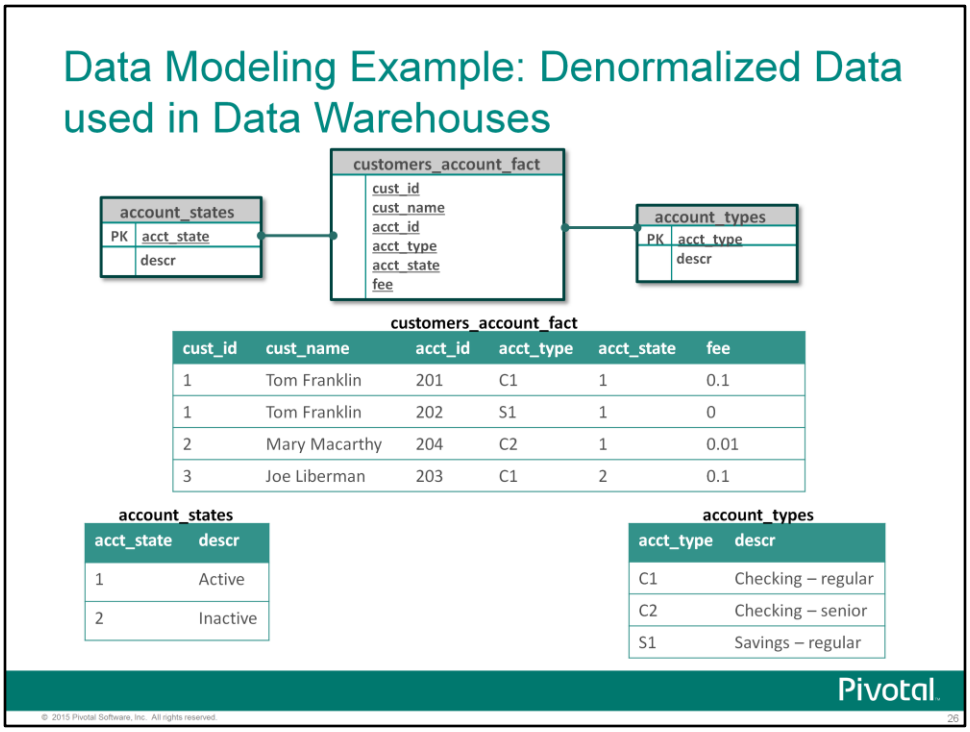


The example shown highlights normalized data, where tables are subject-oriented. Customer information is kept separately from account information, with additional descriptor tables available to provide definition for specific columns in these tables.

Highly normalized data eliminates redundancies, requiring multiple joins to access specific information. While this model is ideal for transactional systems, it introduces long response times when used to generate reports. Data warehouses are used to generate large reports based on millions of rows. Joining tables together where multiple tables are large introduces long processing time.

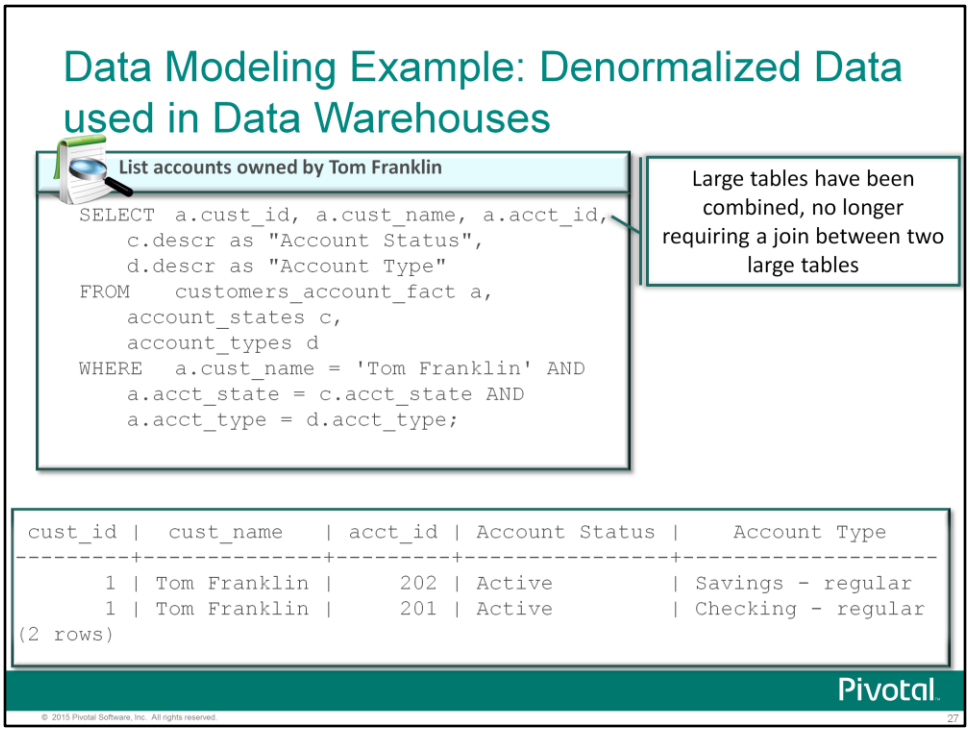


The example highlights the number of joins that are required to extract information on the types of accounts that Tom Franklin owns. It requires that the account and customer tables are joined, along with any other definition tables. The definition tables tend to be smaller, but the accounts and customer tables will be large.



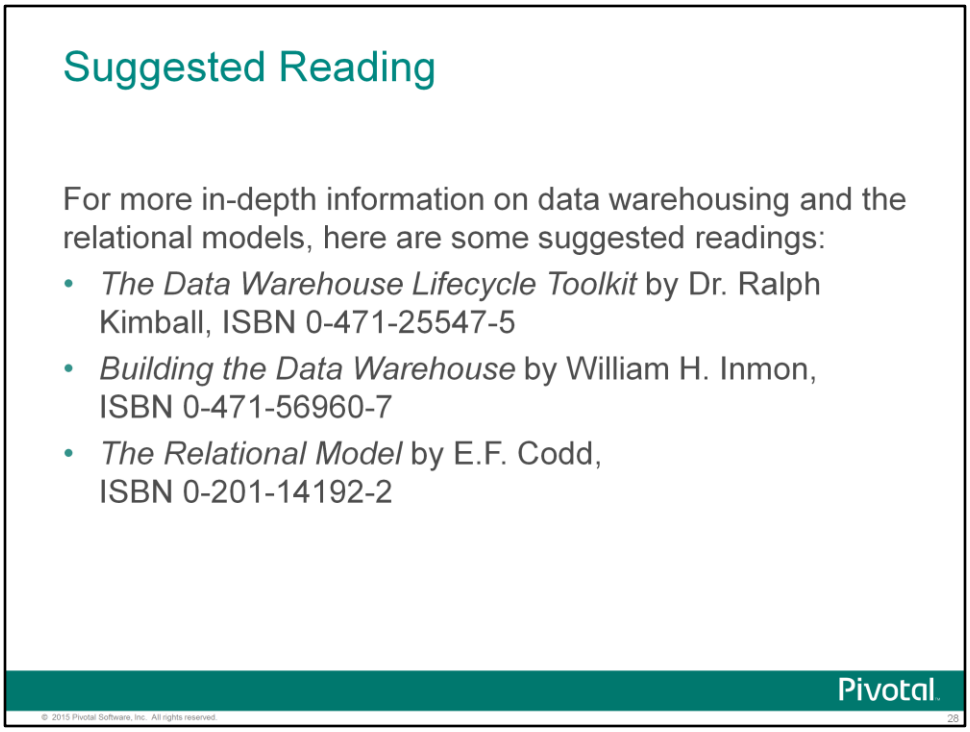
When denormalized, data is grouped together to form tables with larger number of columns. The purpose of this is to optimize the read performance of the database. It potentially introduces redundant data, but reduces the needs to perform joins across multiple tables, particularly large tables.

In this example, the customer and account tables have been merged into a single fact table to reduce the number of joins required to retrieve data from the database. The fee column has also been added to the `customers_account_fact` table because it is a measurement that could change over time. The definition tables that describe aspects of the data are now known as dimensional tables. By combining data into a single fact table, there is a greater chance that the data is physically stored together, reducing the I/O that can be introduced with scanning across multiple disks.



The act of placing the all related facts into a single fact table means there is no longer a need to perform a join on two potentially large tables. In a data warehouse, data may be redundant as it is duplicated in multiple tables. Some customer information may be pulled into another fact table, if necessary, to reduce the I/O that would result from joining multiple large tables to generate reports. By placing the data into a single table, data is more likely to be physically co-located, reducing the number of I/Os necessary to retrieve that data.

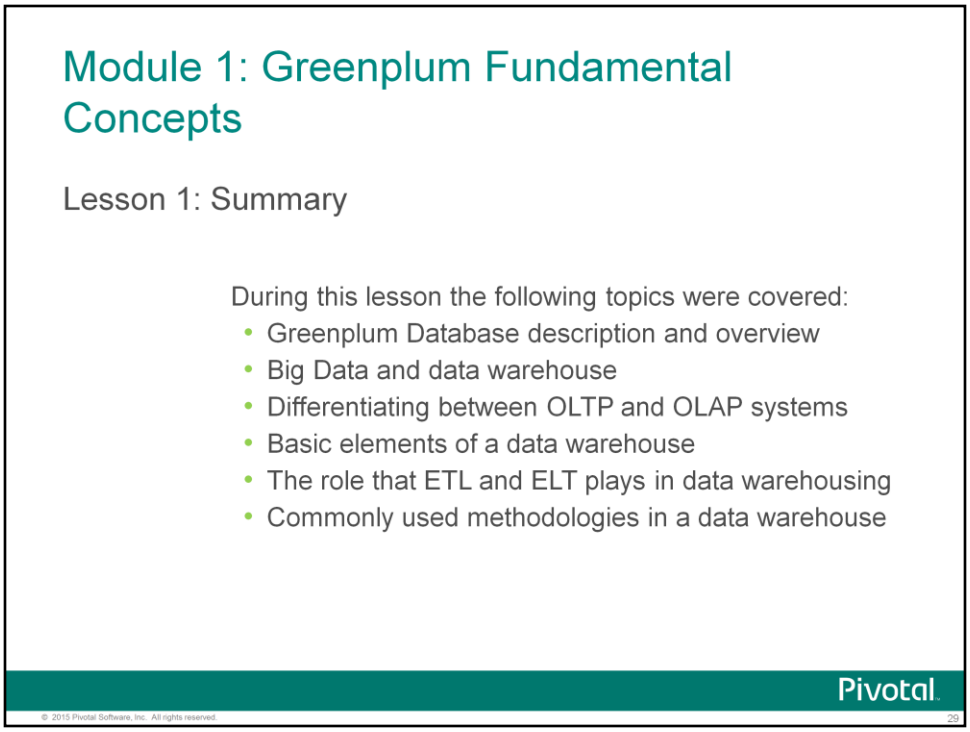
While the example shown here may not reflect the type of information an analyst would be looking to obtain from a data warehouse, the purpose was to highlight the denormalization of data and its potential impact on how queries are written and the speed of the results based on the table structures.



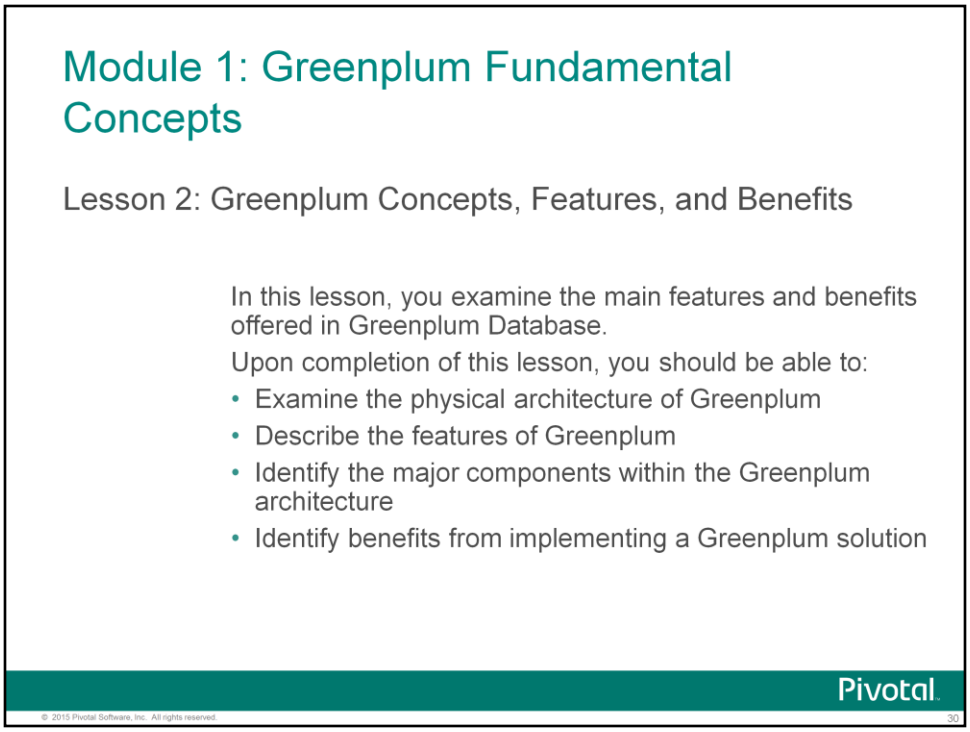
For more information on data warehousing and relational models, reference the following books:

- *The Data Warehouse Lifecycle Toolkit* authored by Dr. Ralph Kimball
- *Building the Data Warehouse*, by William H. Inmon
- *The Relational Model*, by E.F. Codd

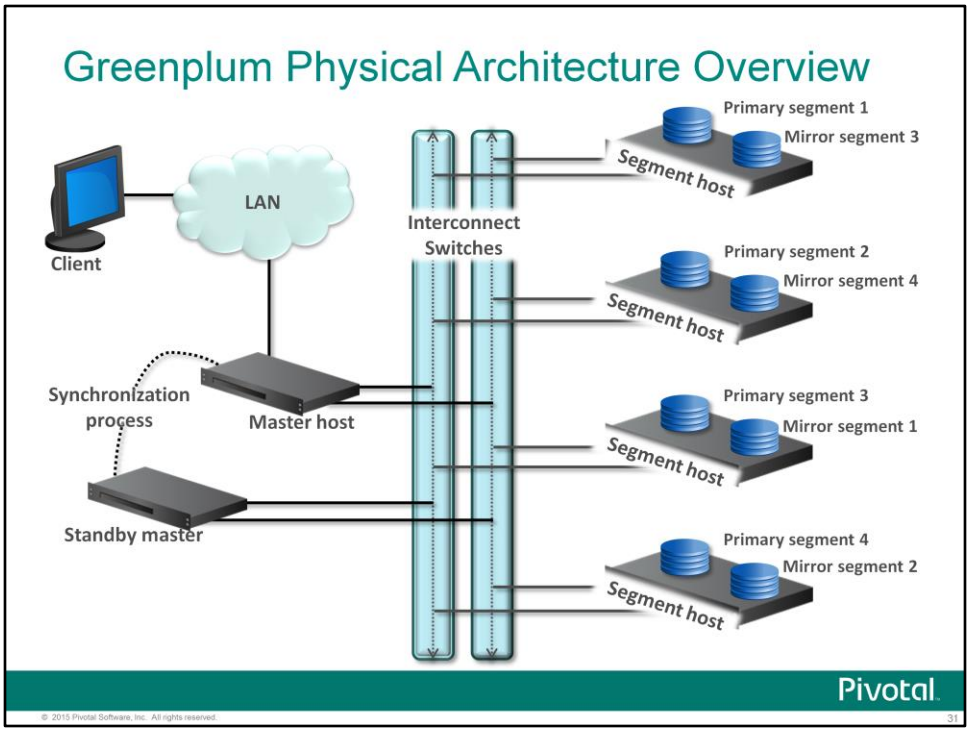
While Dr. Ralph Kimball and William H. Inmon represent two separate camps on the discussion of data warehousing, there is no right or wrong in these ideas. They merely represent two different philosophical approaches.



This lesson provided an overview of the Greenplum Database, along with the overall goal and benefits of the product. Before going into great depths into the database, we examined the concepts of Big Data and the data warehouse, differentiating between OLTP and OLAP systems, a combination of which may be found in many organizations and businesses. The basic elements and components of a data warehouse was defined and the role in ETL compared to ELT was also described. Finally, an overview of the data methodologies used in data warehousing was described at a high level.



In this lesson, you explore the features and benefits offered in Greenplum Database. You also examine the high-level architecture to understand why Greenplum Database successfully handles mission critical analytics.



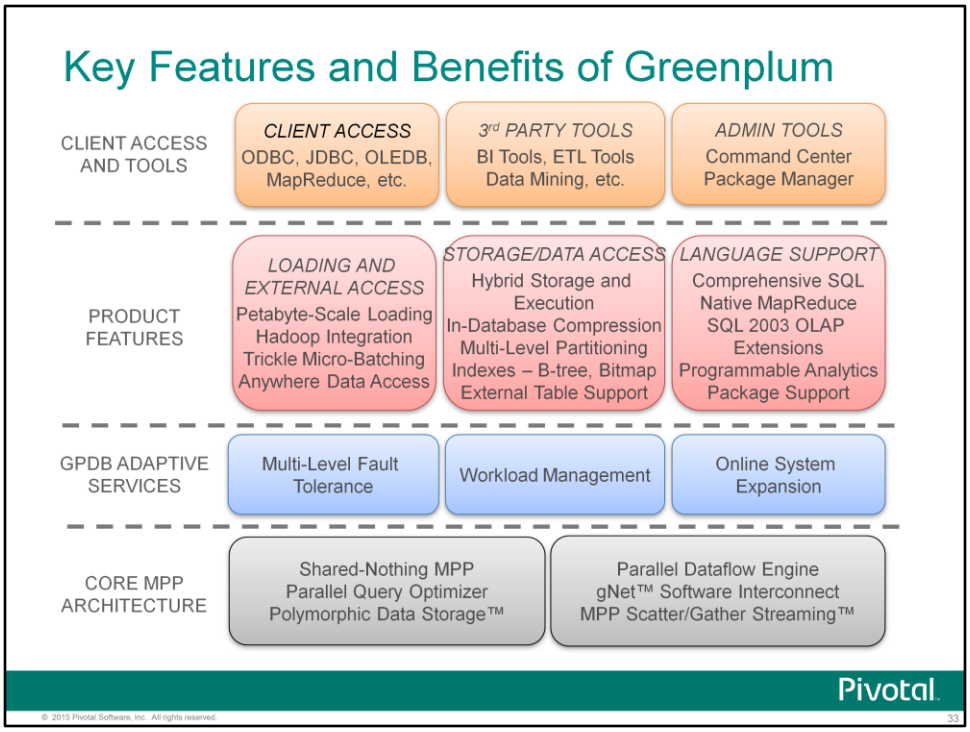
Before examining the features of Greenplum, let us quickly examine the physical architecture of the Greenplum environment and define terminology you will see as you proceed through the course.

A Greenplum environment, or a cluster, consists of the following components:

- **Master host** – The master host is the server responsible for accepting and distributing queries. It is the entry point into the Greenplum environment. While the master itself does not store data, it provides a single abstracted view to all of the data across the segments.
- **Standby master** – The standby master is a warm standby server that is activated when the master host is unavailable.
- **Node or segment host** – The node, or segment host, runs database server processes known as segment instances. It supports one or more segments, depending on the number of processor cores present on the node.

Greenplum Physical Architecture (continued)

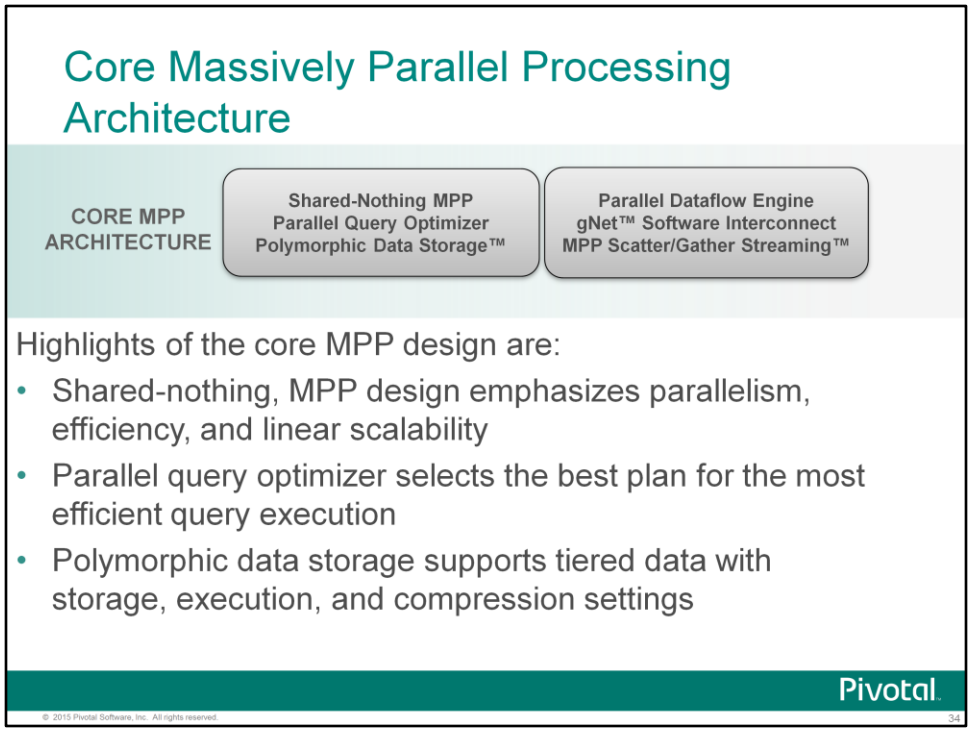
- **Primary Segment** – A primary segment stores a distinct portion of the data and is responsible for handling queries. Depending on your hardware configuration, you may have one or more segments per node. Each time data is loaded into the database, a distribution algorithm is used to determine which segments will store what data. When a query is sent to the master, the master develops a query plan and sends this plan to the segments. Each segment is responsible for executing the query on its set of data.
- **Mirror segment** – The mirror segment is a standby segment that is activated should its corresponding primary segment no longer be available.
- **Interconnect switches** – The interconnect switches are the heart of the communication in the Greenplum environment.



Greenplum is a shared-nothing, massively parallel processing (MPP) architecture designed for business intelligence and analytical processing. It is one of the first open-source databases, based on PostgreSQL, that was made available to enterprise environments. Built to support the next generation of Big Data, Greenplum manages, stores, and analyzes terabytes to petabytes of data, with vastly improved performance over traditional relational database management system products.

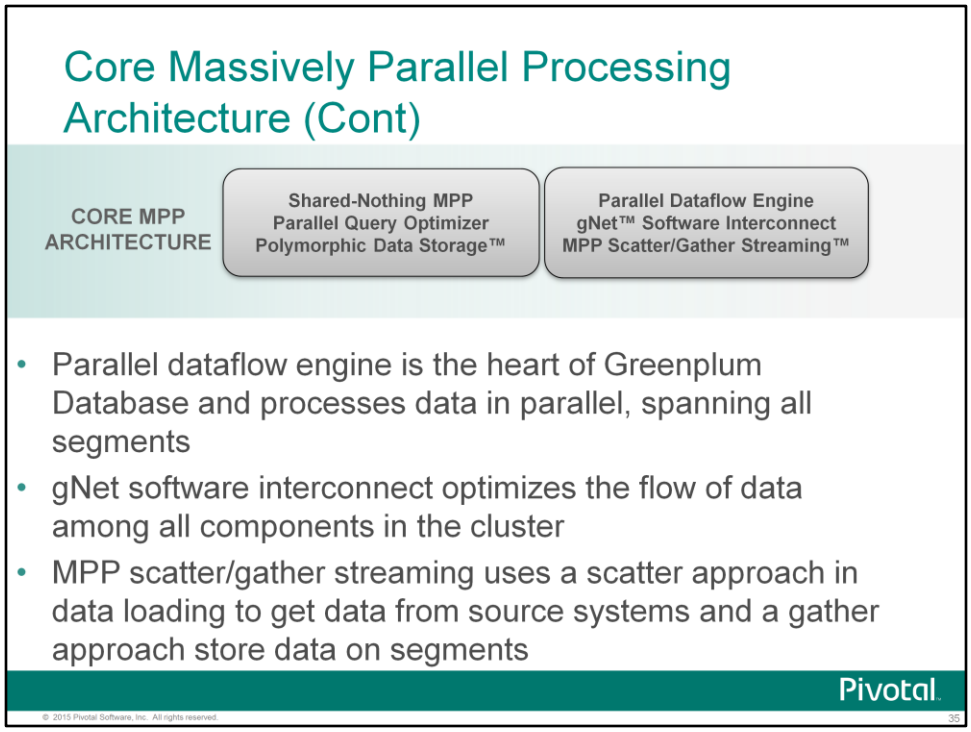
The logical architecture represents the major features and benefits Greenplum offers. Starting from the bottom of the illustration, you have:

- Core massively parallel processing architecture
- Greenplum adaptive services
- Product features
- Client access and tools

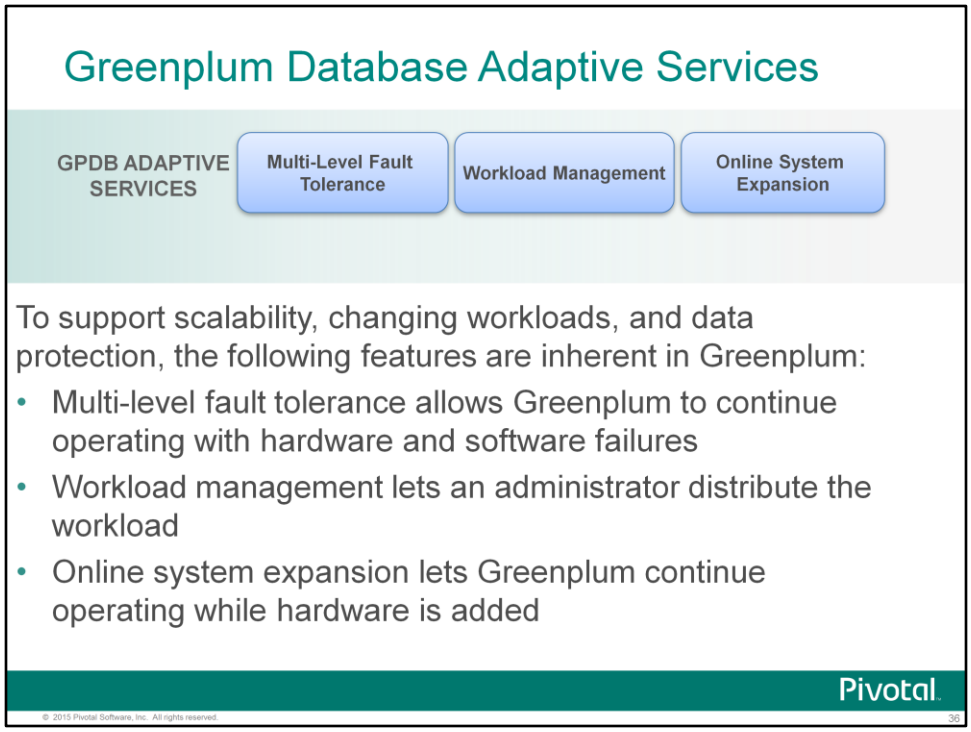


The core massively parallel processing architecture highlights several major features:

- **Shared-nothing, MPP design** – The shared-nothing, massively parallel processing architecture utilized by Greenplum incorporates the highest level of parallelism and efficiency to handle complex business intelligence and analytical processing. Greenplum takes advantage of the available hardware in its environment to ensure that data is automatically distributed and query workload is parallelized. Each unit, or segment, within the environment, acts as a self-contained database management system that owns and manages a distinct portion of the overall data. While the data and execution are parallelized, all nodes within a Greenplum environment work together in a highly coordinated fashion.
- **Parallel query optimizer** – When it receives a query, the master server uses a cost-based optimization algorithm to evaluate a vast number of potential plans and selects the one it believes is the most efficient. It does this by taking a global view of execution across the cluster and factors in the cost of moving data between nodes. By taking a global view, you obtain more predictable results than an approach which requires replanning at each node.
- **Polymorphic data storage** – A polymorphic data storage allows customers to choose the storage, execution, and compression settings to support row or column oriented storage and retrieval. This lends support to tiered or temperature aware data, where you may opt to store older data as column oriented with deep archival compression on one set of disks, while more recent data is stored with fast and light compression.

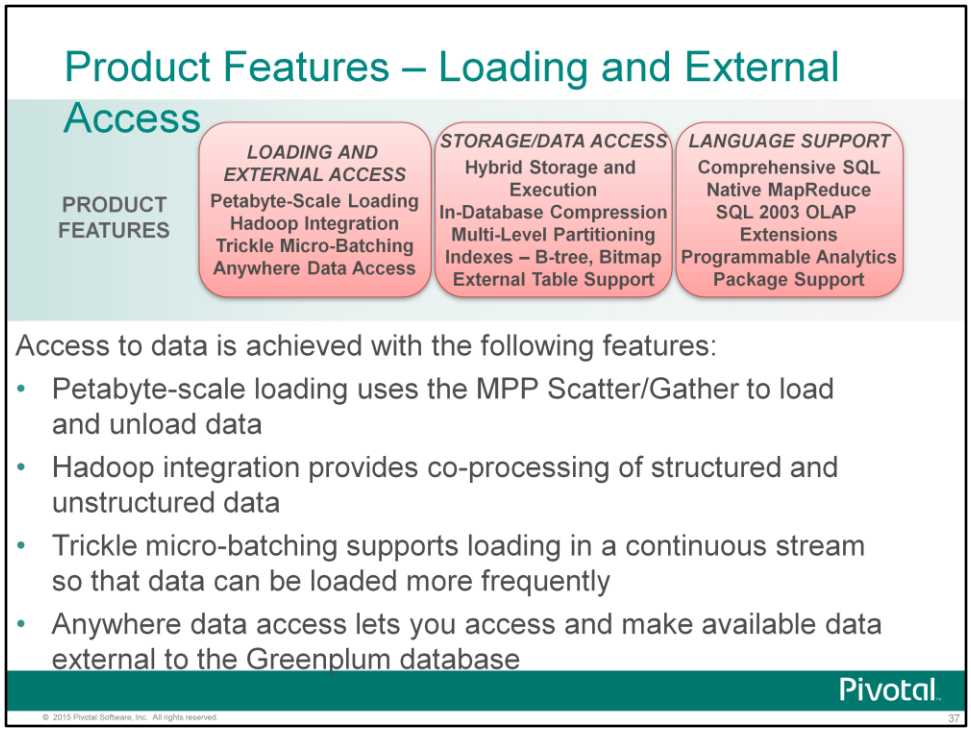


- **Parallel dataflow engine** – The work of processing and analyzing data is performed in the parallel dataflow engine, the heart of Greenplum Database. This optimized parallel processing infrastructure processes data as it flows from disk, external sources, or other segments over the interconnect. The engine is inherently parallel and spans all segments of a cluster. It can effectively scale to thousands of commodity processing cores.
  It is highly optimized at executing both SQL and MapReduce in a massively parallel manner. It has the ability to directly execute all necessary SQL building blocks, including performance-critical operations such as hash-join, multistage hash-aggregation, SQL 2003 windowing, and arbitrary MapReduce programs.
- **gNet software interconnect** – One of the most critical components in Greenplum, the gNet interconnect optimizes the flow of data to allow for continuous pipelining of processing without blocking on all nodes of a system. It leverages commodity Gigabit Ethernet and 10GigE switch technology to efficiently pump streams of data between motion nodes during query plan execution. It utilizes pipelining, the ability to begin a task before its predecessor task has completed, to ensure the highest-possible performance.
- **MPP scatter/gather streaming** – Using the MPP scatter/gather streaming, Greenplum is able to achieve data loads of more than 4 terabytes per hour with negligible impact on concurrent database operations. Using a parallel-everywhere approach to data loading, data is scattered from all source systems across hundreds or thousands of parallel streams to all nodes I the cluster. Each node in the cluster simultaneously gathers the data it is responsible for.



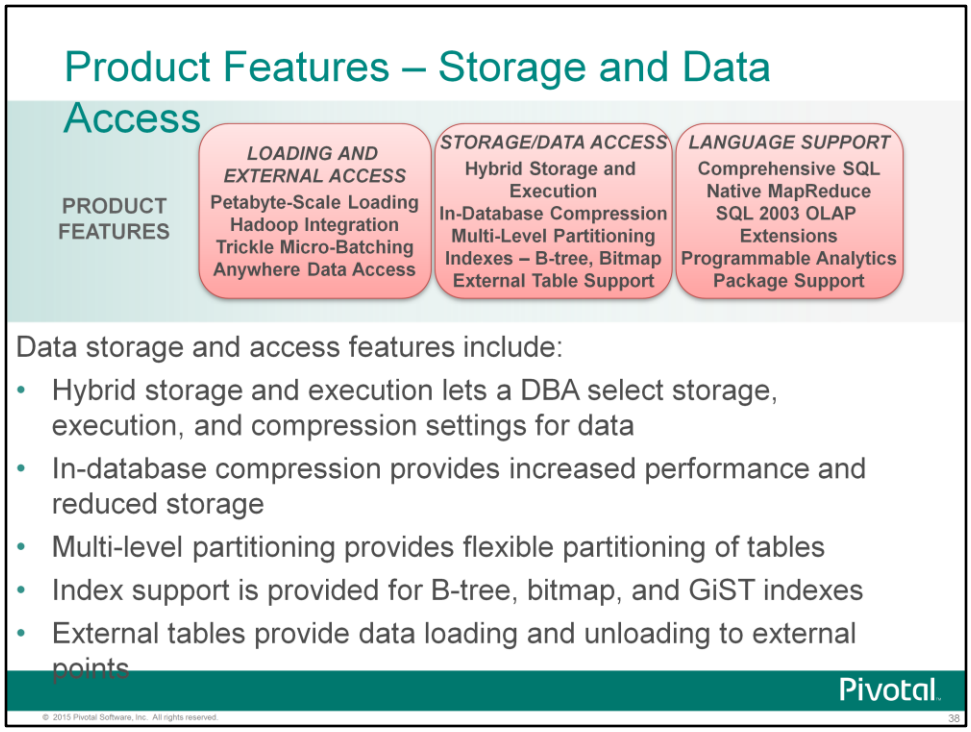
Scalability, workload management, and fault tolerance are features that allow Greenplum to adapt to a changing environment, increase uptime, and scale storage and compute power.

- **Multi-level fault tolerance** – Using multiple levels of fault tolerance and redundancy, Greenplum can continue operating in the face of hardware and software failures. Mirrors for the master and segments help to protect against data loss as well as database operation loss. The interconnect provides redundant access to all nodes, the master, standby master, and any other components connected to the switch.
- **Workload management** – The database administrator has administrative control over determining system resources to users and queries. User-based resource queues automatically manage the flow of work to the databases from defined users. Query prioritization allows control of runtime query prioritization to ensure queries have appropriate access to resources. This allows you to prevent one query from hogging all system resources and potentially starving other queries out of these resources. This also allows you to redistribute resources based on user loads.
- **Online system expansion** – Servers can be added to not only increase storage capacity, but also to increase processing power and loading performance. The database can remain online while the expansion process takes place in the background. Due to the implementation of the shared nothing, MPP design, increasing the number of nodes in the cluster increases performance and capacity linearly for Greenplum. Support for dynamic provisioning means you can add onto existing configurations without having to replace existing configurations.



To load data and access external data, Greenplum offers the following features:

- **Petabyte-scale loading** – Using the MPP Scatter/Gather streaming technology, Greenplum can perform high-performance loading and unloading of data. With each additional node in the cluster, the speed at which the loads, parallel data digest, and unloads, parallel data output, are performed increases linearly.
- **Hadoop integration** – Hadoop provides a solid method of processing unstructured data. Greenplum Database provides high performance parallel import and export of data from Hadoop clusters and filesystems.
- **Trickle micro-batching** – When loading a continuous stream, trickle micro-batching allows data to be loaded at frequent intervals, such as every five minutes, while maintaining extremely high data ingest rates. This is extremely useful to companies such as the NYSE or other trade and securities institutions.
- **Anywhere data access** – Data external to the Greenplum database can be accessed, regardless of their location, format, or storage medium. Greenplum allows you to define external tables that access this data and makes it available for reads or writes.

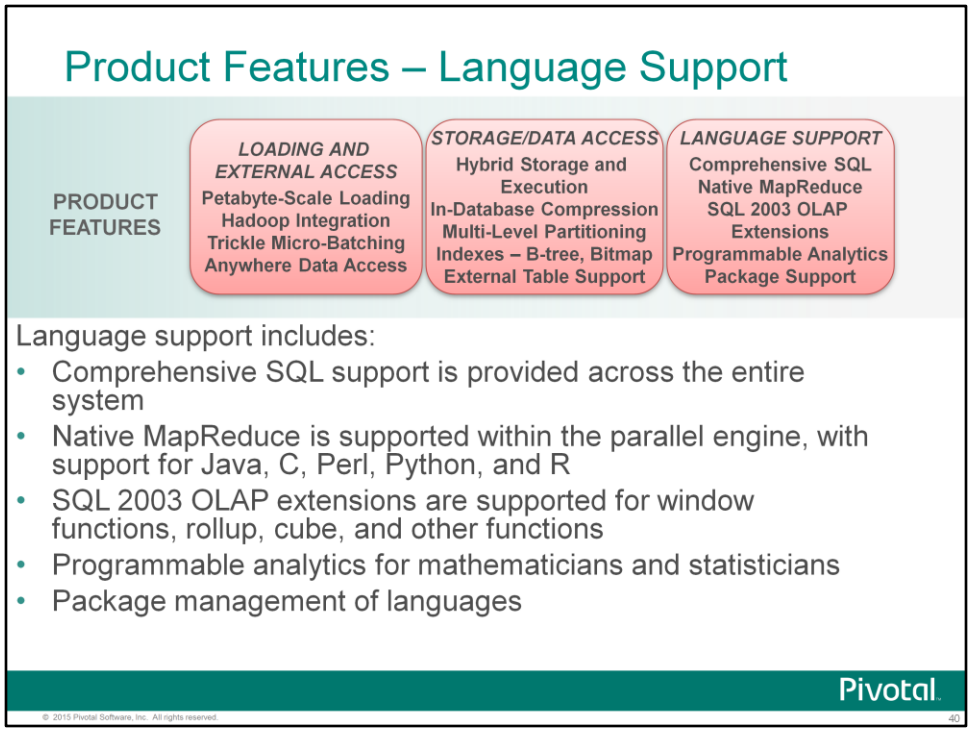


Data storage and access features include:

- **Hybrid storage and execution** – For each table or partition of a table, the database administrator can select the storage, execution, and compression settings that suit the way that table will be accessed. This feature includes the choice of row- or column-oriented storage and processing for any table or partition. This leverages Greenplum's Polymorphic Data Storage technology and allows for tiered storage, where the database administrator can define which data will have lighter compression to allow for faster access and which is not accessed as frequently.
- **In-database compression** – Increased performance and reduced storage can be achieved with in-database compression. By reducing the amount of disk space data takes up, you see an increase in effective I/O performance. In-database compression allows for the storage of years of data, economically. This allows you to get into a true discussion of compliance, e-discovery, and regulatory issues, where you can pull data from previous years quickly. You may not be able to query as quickly, depending on your storage plan, but you will be able to more quickly access data that hasn't been moved off to slow storage or tape.
- **Multi-level partitioning** – With multi-level partitioning, you have flexible partitioning of tables based on date, range, or value. Partitioning is specified using DDL and allows an arbitrary number of levels. The query optimizer will automatically prune unneeded partitions from the query plan.

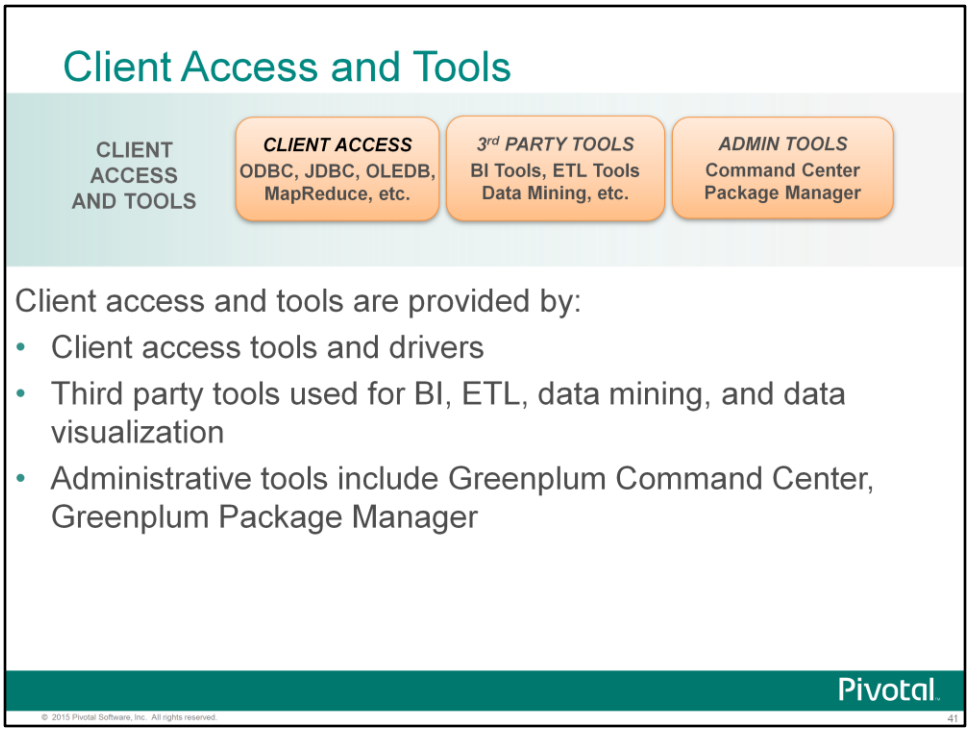
Product Features – Storage and Data Access (continued)

- **Index support** – Greenplum provides support for a range of index types, including B-tree, bitmap, and Generalized Search Tree (GiST) indexes.
- **External Tables** – External tables lets administrators load data into the Greenplum Database using externally available files, web sites, and pipes. Data is unloaded in the same way to external resources. This extends the methods available for loading and unloading data to and from Greenplum Database.



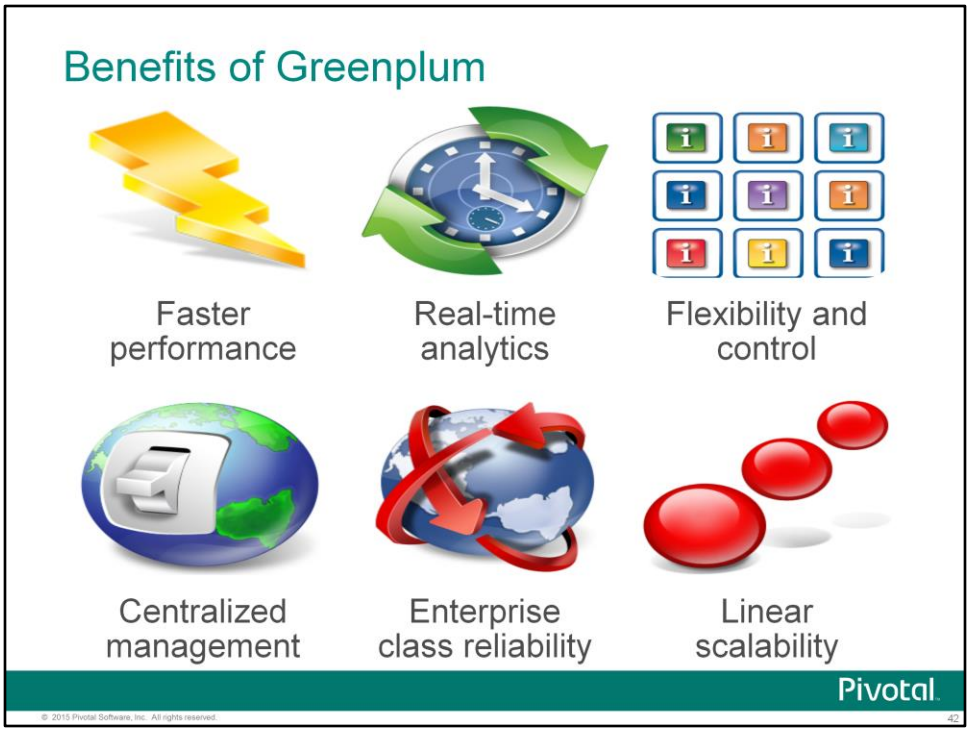
Powerful language support gives developers flexibility in how to approach Greenplum. Language support is provided for:

- **Comprehensive SQL** – Greenplum offers comprehensive SQL-92 and SQL-99 support with SQL 2003 OLAP extensions. All queries are parallelized and executed across the entire system.
- **Native MapReduce** – MapReduce has been proven as a technique for high-scale data analysis by Internet leaders such as Google and Yahoo. Greenplum Database natively runs MapReduce programs within its parallel engine. Languages supported include Java, C, Perl, Python, and R.
- **SQL 2003 OLAP Extensions** – Greenplum provides a fully parallelized implementation of SQL recently added OLAP extensions. This includes full standard support for window functions, rollup, cube, and a wide range of other expressive functionality.
- **Programmable Analytics** – With programmable analytics, Greenplum offers a new level of parallel analysis capabilities for mathematicians and statisticians, with support for R, linear algebra, and machine learning primitives. Greenplum also provides extensibility for functions written in Java, C, Perl, or Python.
- **Package Support** – Greenplum also incorporates package support to provide turn key analytic extensions that allows you to more easily manage your language extensions.



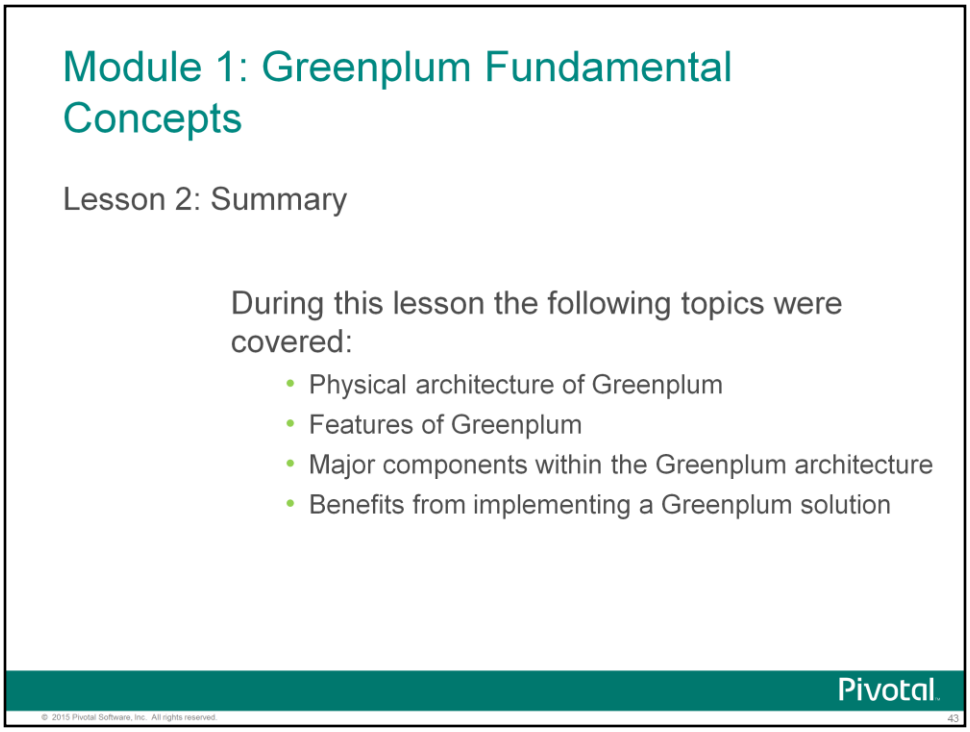
Users and administrators have access to Greenplum through a variety of tools, including:

- **Client access** – Tools and drivers such as ODBC, JDBC, and OLEDB, can be used to access the Greenplum database. MapReduce lets you process large data sets with a parallel distributed algorithm.
- **Third party tools** – Business intelligence tools, ETL tools, applications for data mining and data visualization can also gain access to the Greenplum database. Health monitoring and alert notifications are provided with email and simple network management protocol (SNMP) notifications.
- **Administrative tools** – Greenplum Command Center lets administrators manage and monitor the state of the system and workloads, including system metrics and query details on the system. Command Center provides a dashboard for managing and monitoring the system and database, along with queries. You can drill down into a query's detail and plan to understand its performance.
  Greenplum Package Manager lets you install additional supported languages through a package management utility.
  You can access your environment with tools such as such as pgAdmin 3. pgAdmin 3 is the most popular and feature-rich Open Source administration and development platform for PostgreSQL.

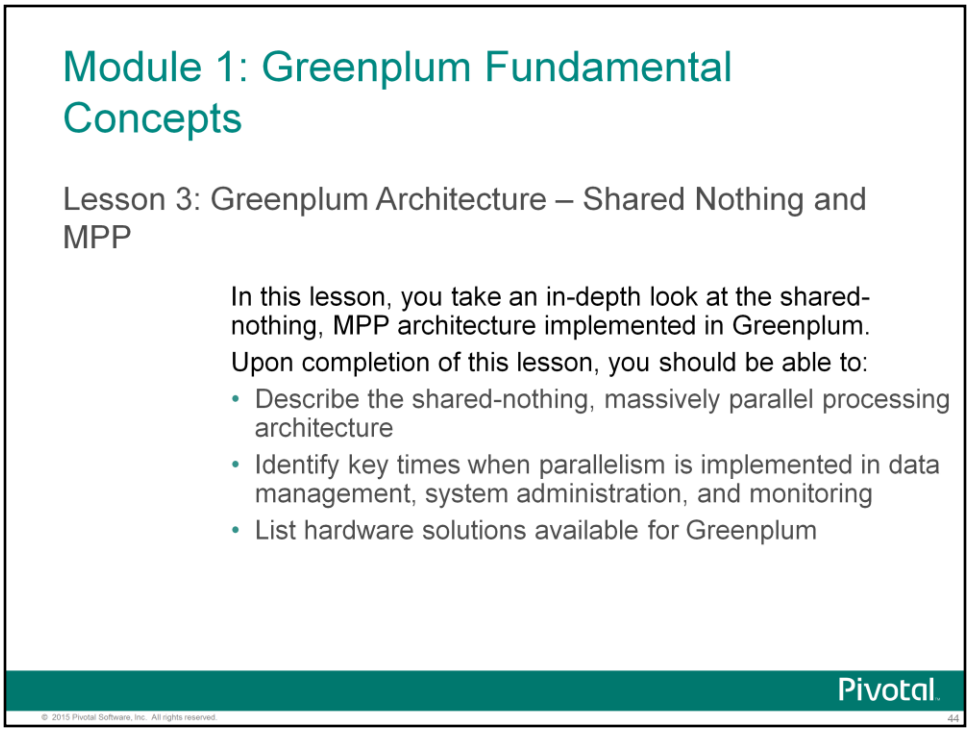


Customers who implement Greenplum gain benefits in:

- **Faster performance** – Increased resources support increased workload. You can see performance gains of 10 to 100 times faster than traditional data warehouse technologies.
- **Real-time analytics** – Greenplum enables sophisticated queries and ad-hoc analysis with multiple terabytes to petabytes of data. With OLAP queries, you can perform advanced queries without having to use third party tools.
- **Flexibility and control** – You can decide whether to choose EMC's hardware or your own. This gives you control over the choice of hardware and operating systems, as well as the ability to add capacity and therefore performance, inexpensively.
- **Centralized management** – Centralized management allows ease of configuration through a single central location – the master. In the end, centralized cluster management and administration lowers total cost of ownership (TCO).
- **Enterprise class reliability** – High availability, mirroring on segments and standby and hardware-level mirroring. With multiple levels of redundancy and fail-over, this minimizes downtime.
- **Linear scalability** – Nodes can be expanded on an as needed basis. This allows for predictable, linear performance gains and capacity growth. It is recommended that Professional Services is involved when expanding nodes. Expanding nodes involves reconfiguring the database to make immediate use of the hardware with preexisting data and newly stored data.

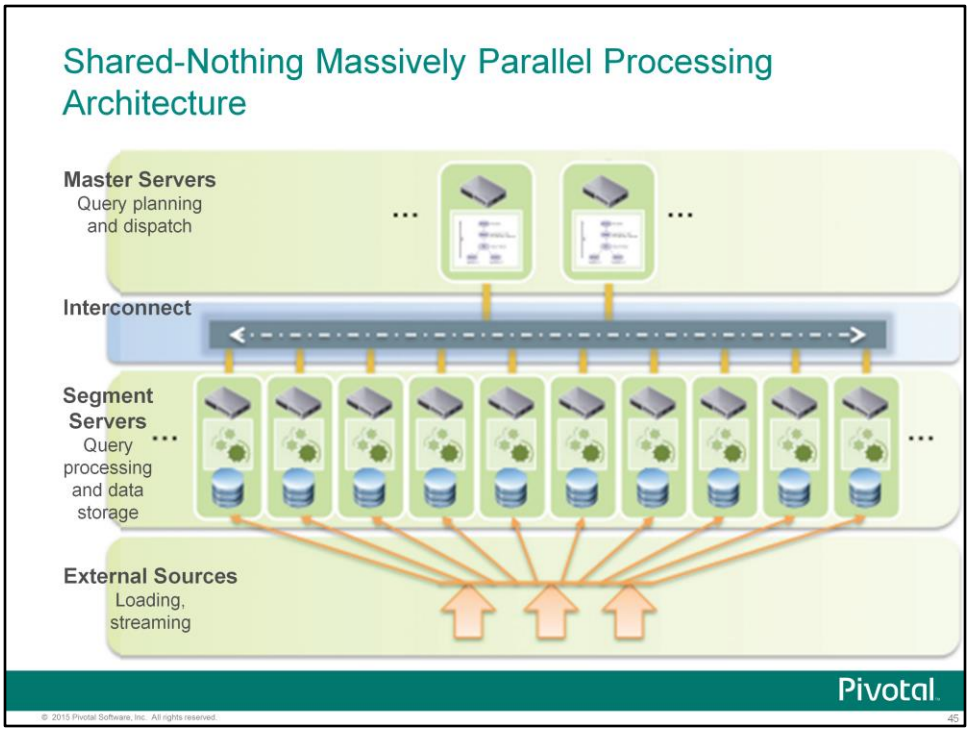


This lesson provided a more in-depth description of the Greenplum Database, including the physical architecture that makes up the database, the major features of the Database, the components within the architecture and how they interact with each other and clients, and the benefits from implementation a Greenplum Database solution within your environment.



In this lesson, you examine the shared-nothing, MPP architecture in greater detail, analyzing how and where parallelism is implemented in Greenplum.

You will also examine the type of hardware solutions that are available for the Greenplum Database.



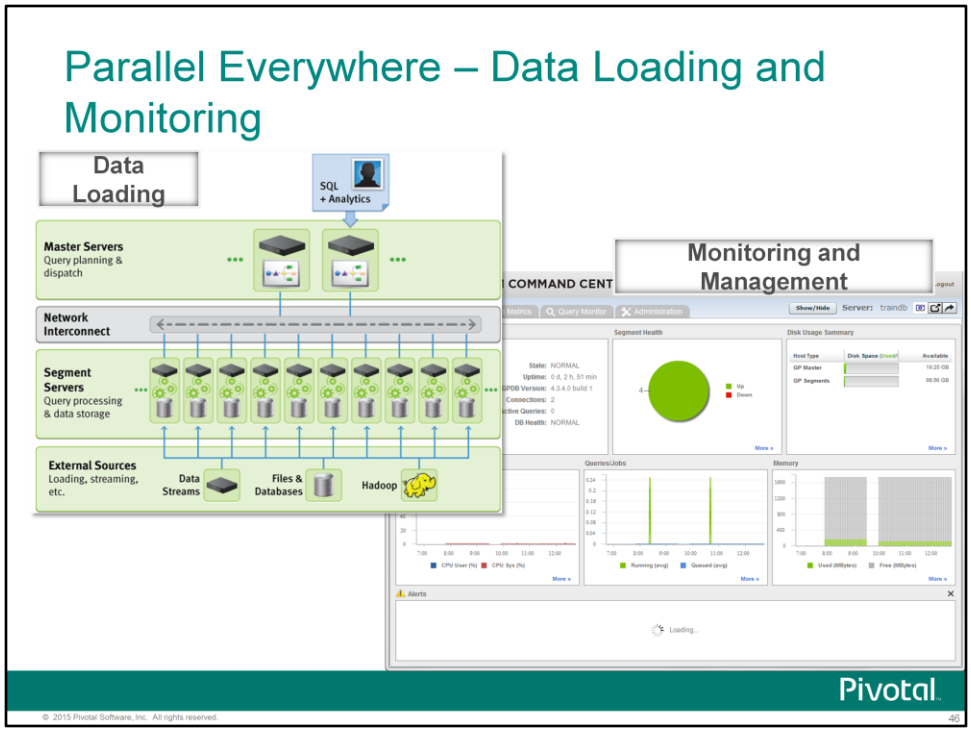
The shared-nothing, massively parallel processing architecture refers to systems with two or more processing units, or segment hosts, which cooperate to carry out an operation. Each segment host has its own:

- Processor
- Memory
- Disk
- Operating system

Each segment host runs its own Greenplum database instance. Greenplum leverages this high performance system architecture to distribute the load of multi-terabyte data warehouses. It uses all of a system's resources in parallel to process a query. In a sense, Greenplum Database is a cohesive database management system comprised of several PostgreSQL database instances acting together.

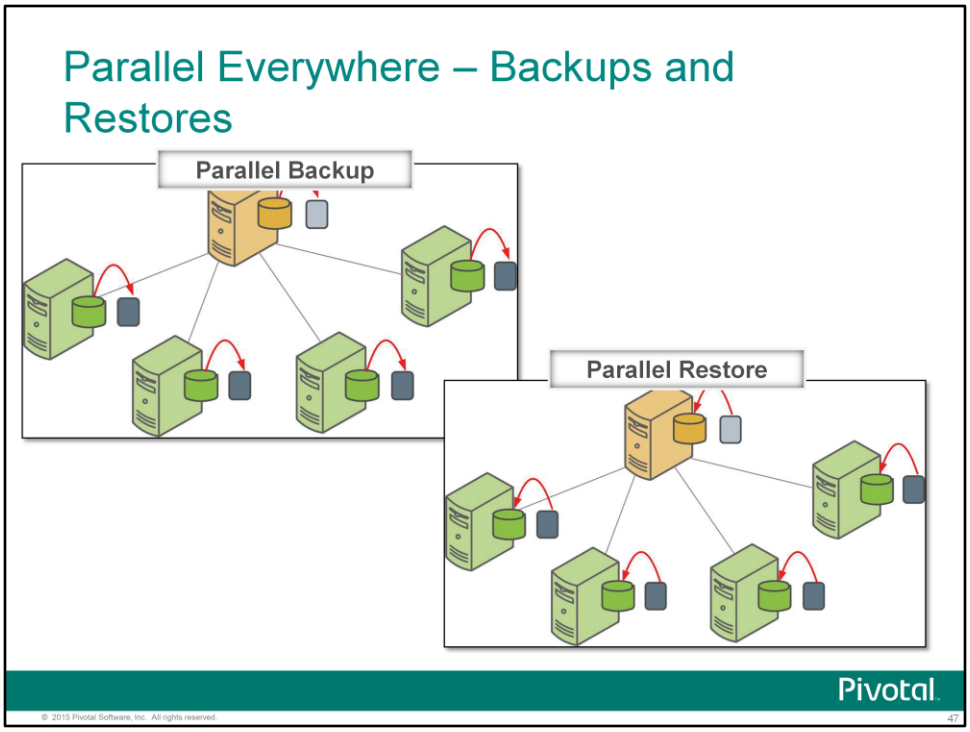
While Greenplum is based on PostgreSQL 8.2.9, maintaining SQL support, main PostgreSQL features, configuration options, and the same end-user functionality, it extends PostgreSQL by including features for business intelligence workloads, including:

- Parallel data loading
- Implementation of external tables
- Resource management
- Query optimizations
- Storage enhancements



In the shared-nothing, MPP architecture, parallelism is key to the following:

- **Data loading, storage, and querying** – Data loading into the Greenplum databases can come directly through an ETL host connected to the interconnect. The `gpfdist` command, provided with Greenplum, is a custom command that runs an internal HTTP light server and allows Greenplum to connect to the ETL. Data is loaded to all segments simultaneously, using the Scatter/Gather method.
  The query execution plan is broadcast to all segments, even if they do not contain data. The segments then respond to the query plan with appropriate data.
  All of the work is performed in parallel: segments work to load, read, and modify the user data in parallel. While some segments may not contain data, the query plan is still issued to all segments.
- **Monitoring and Management** – When monitoring the Greenplum environment, Greenplum Command Center checks the status of the system as a whole. You can obtain the status of each individual Greenplum instance in the cluster, while managing the database and checking on the performance of queries executing in the database. Command Center also lets you manage specific aspects of the database. This includes but is not limited to controlling queue priorities, resource queue management, and database status and availability.



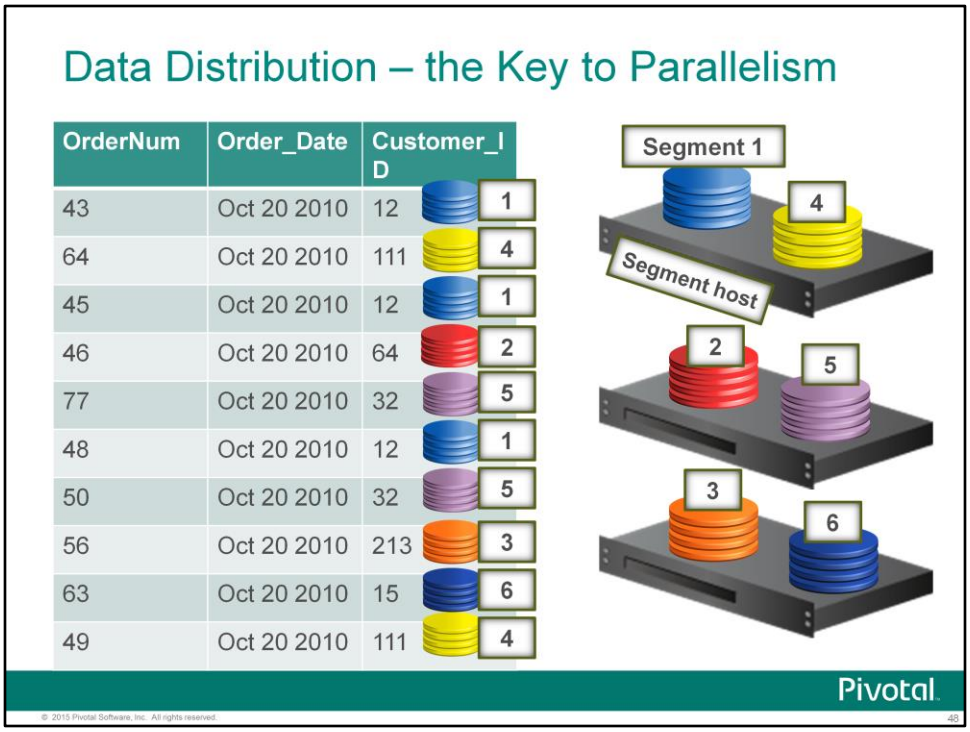
Backup and restore is another function performed in parallel. When you issue the command to backup data, data is compressed and archived onto the appropriate host in parallel. Restores to segments are also retrieved in parallel.

During backup, data is backed up to a local backup directory. The backup is archived in parallel. Restore functions in the same way, retrieving in parallel.

A Data Domain Device is an optimal solution that can be used for backup and restores. Data is backed up, compressed, and moved onto an archive host. Restores are retrieved from the archive host and then uncompressed.

The key to performing a backup is to have enough local storage on the segment host and master host to perform the backups. While Greenplum itself is not necessarily focused on backups, it works well with other software, such as Data Domain, to provide data backups, archival, and restoration services.

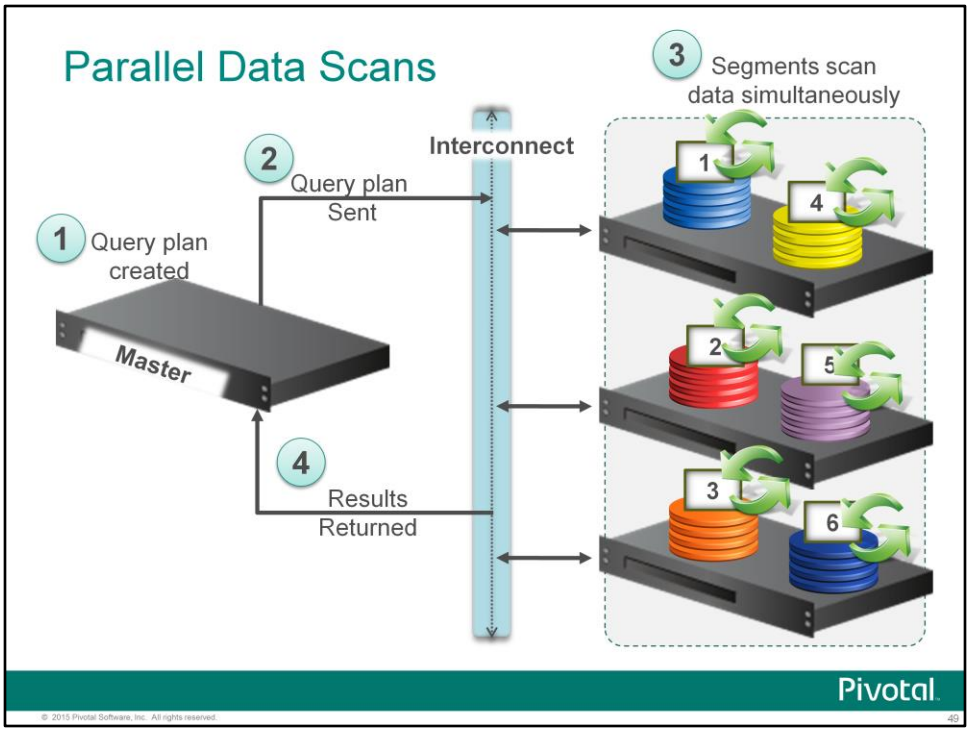
All backups and restorations are performed on the master.



When distributing data, the best scenario is to have the data co-located as much as is possible. The data should be spread as evenly as possible across as many segments as possible. There are several key points to remember when distributing the data:

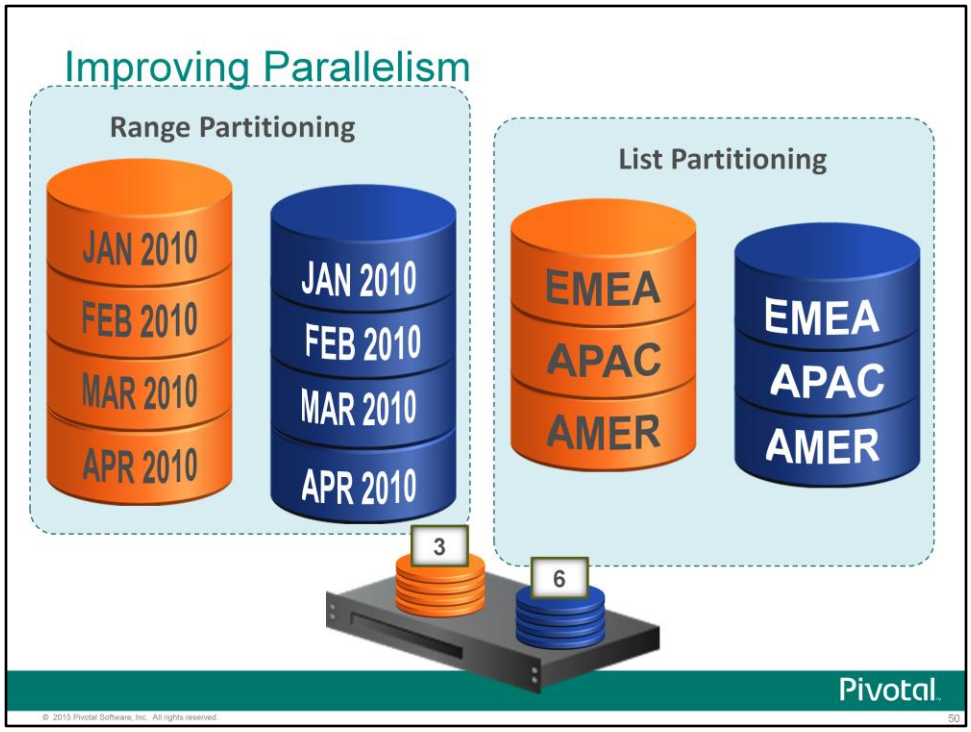
- The distribution key is used to determine how the data is spread across the segments. Choose the distribution key wisely.
- The data type is important for hashing out the distribution policy used for distributing the data. An integer and a BigInt do not hash in the same manner.
- Analyze the data and examine commonly run queries. This helps to determine which column should be used as the distribution key.

In the example shown on the slide, data is distributed on the distribution key, `Customer_ID`. Choosing `Customer_ID` as the distribution key more evenly spread the data across all segments. If you had chosen the `Order_Date` column, all of the data, based on this sample set, would be written to a single segment.



When a query is sent to the master server:

1. The master server develops a query plan by taking a global view of execution across the entire cluster and factors in any movement that must occur, such as when joining tables together. This helps to create a more predictable result.
2. The query plan is sent to all of the segments within the cluster.
3. Each segment scans its set of data independently of other segments. This action occurs in parallel.
4. Once all segments have completed executing the query based on the query plan, the results are returned to the master, which in turn returns the results to the client.



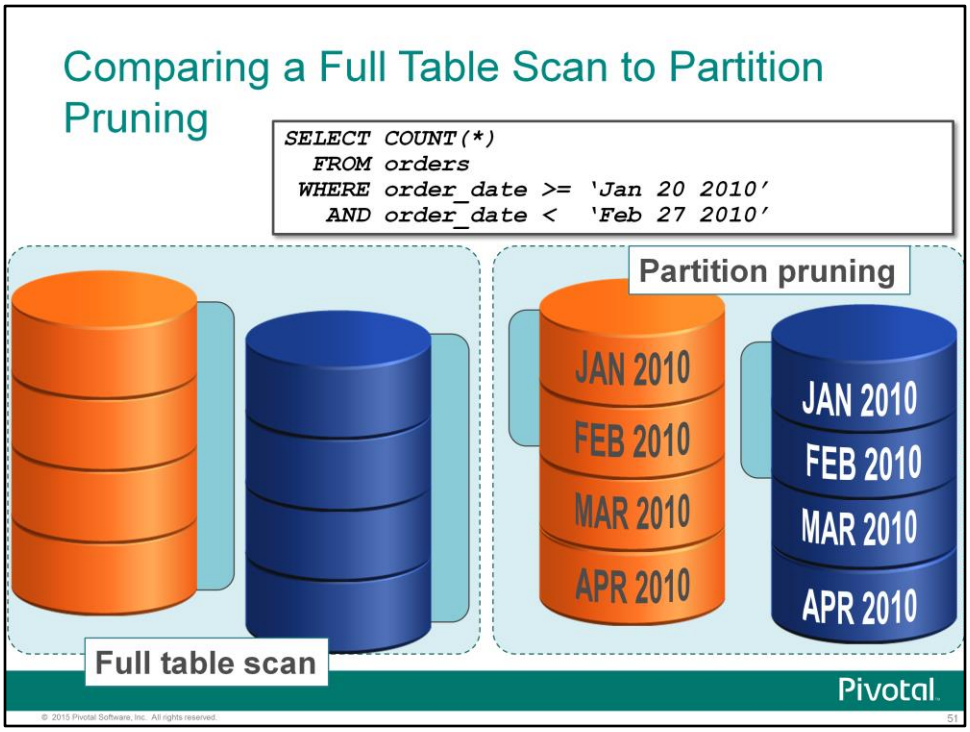
To improve the results achieved with parallelism, Greenplum supports data partitioning.

Partitioning large tables can reduce the number of rows that needs to be scanned when performing a data scan for a query. Data is broken into bucket on each segment, based on a filter. When a data scan is performed, only the buckets that match the query are searched. This reduces the scan size and therefore the work that each segment must perform to respond to a query.

Two types of partitioning are supported:

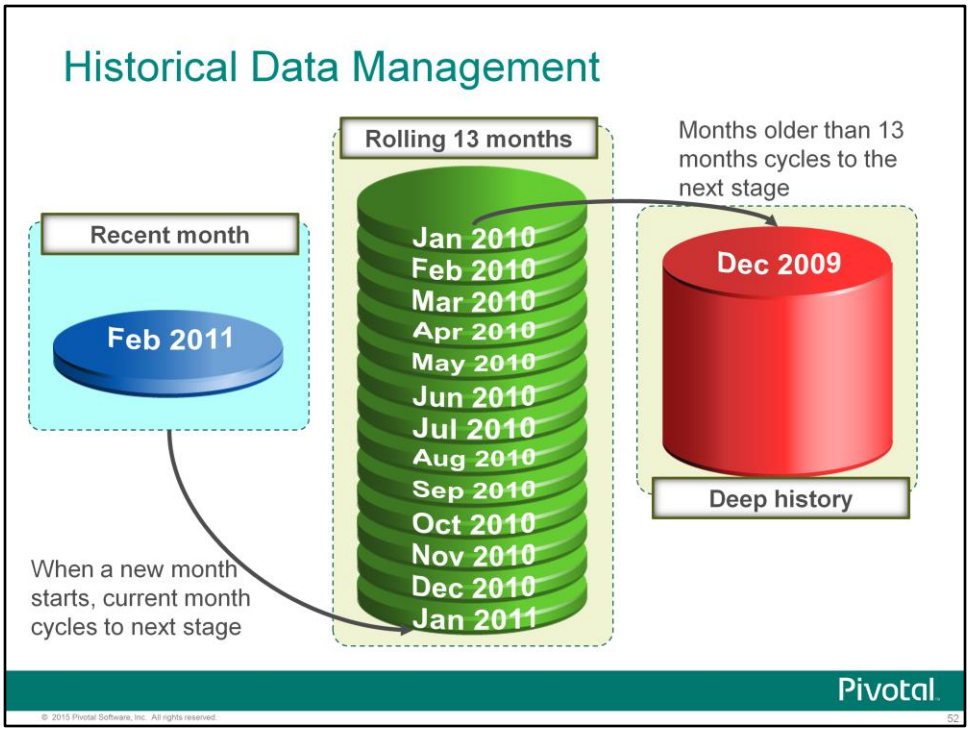
- **Range partitioning** – Data is partitioned based on a numerical range, such as date or price.
- **List partitioning** – Data is partitioned based on a list of values, such as sales territory or a product line.

Note: Partitioning is above and beyond the hash-based distribution and allows the system to scan just the subset of buckets that might be relevant to the query



When presented with a query, each segment scans its data to yield the appropriate results. If the table is not partitioned, a full table scan is performed.

If the table is partitioned, only the buckets that match the criteria are scanned. This can greatly reduce the amount of work that must be performed by each segment.

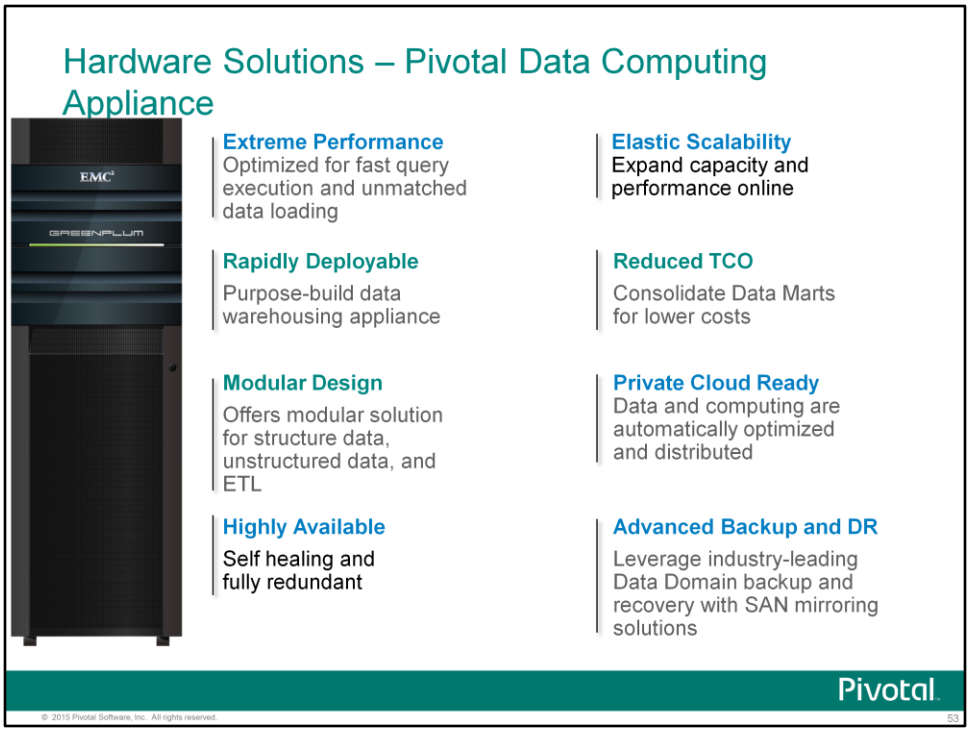


Partitions do not have to be the same size or orientation, which can be column or row-oriented. You can partition your data based on a selected time frame.

This example shows a rolling management scheme, where 13 months of data are maintained at a time. The scheme is as follows:

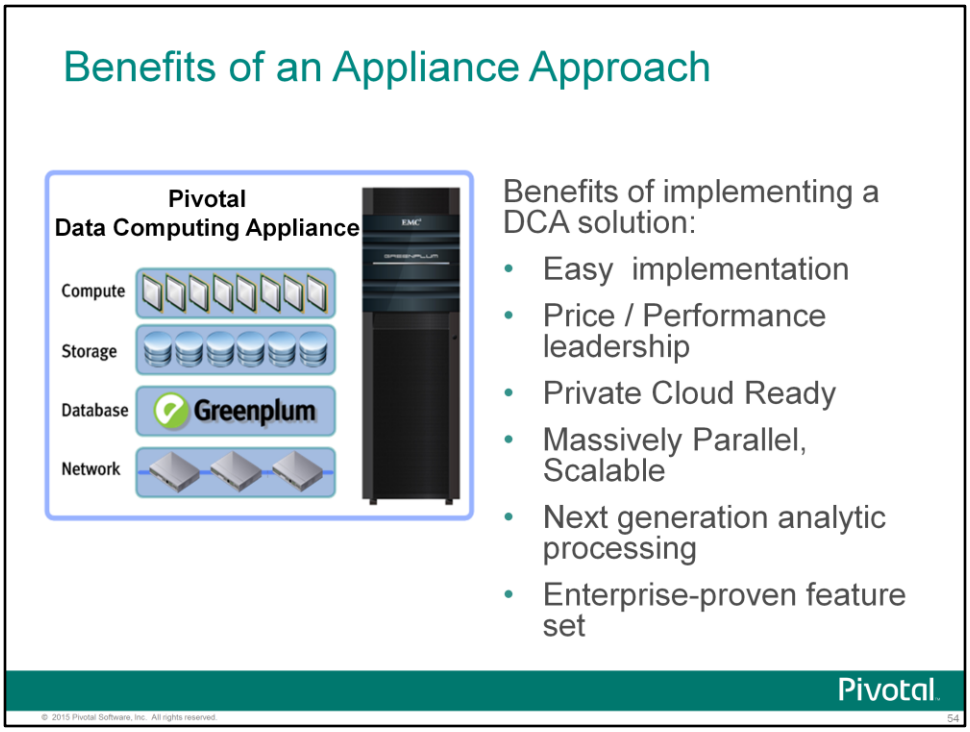
- Anything older than 13 months is moved to deep history.
- The current month is maintained *in memory* where access is fastest.
- All in between months are maintained at a second level of storage

The strategy is customizable based on your organization's needs.



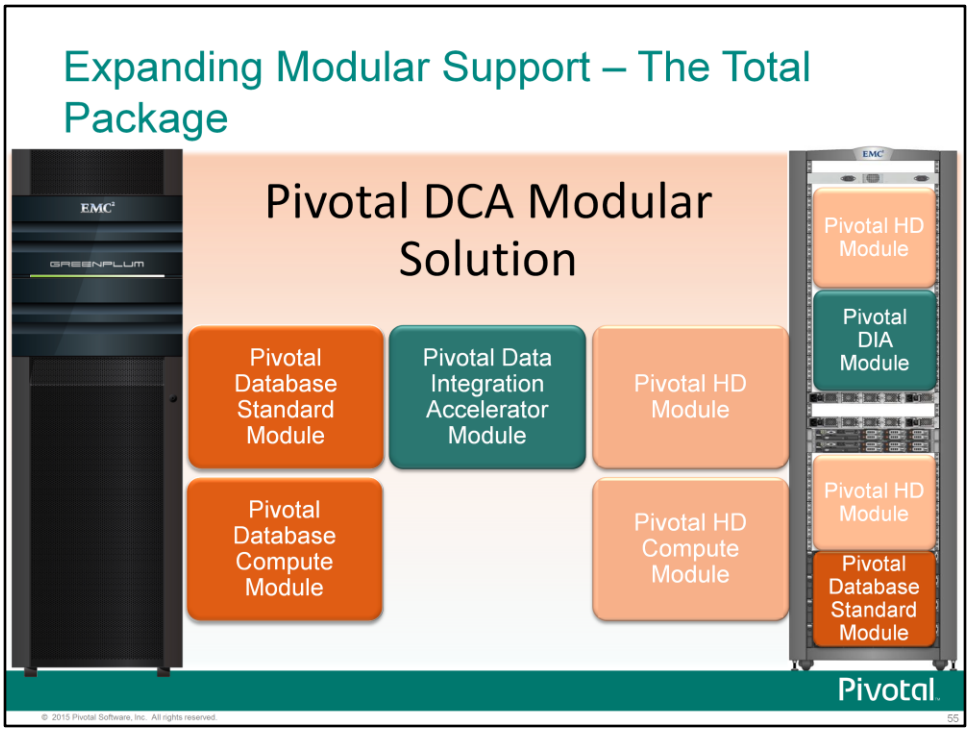
The Pivotal Data Computing Appliance (DCA) is a purpose-built, highly scalable parallel data warehousing appliance that architecturally integrates database, compute, storage and network into an enterprise class easy-to-implement system. It offers a modular solution for structured data, unstructured data, and Greenplum partner applications for Business Intelligence and ETL services. The Pivotal DCA:

- Brings the processing power of an MPP architecture
- Delivers the fastest data loading capacity, and the best price/performance ratio in the industry
- Does not have the complexity and constraints of proprietary hardware
- Provides a flexible infrastructure to support expansion based on growing business needs



While you can implement any reference architecture within your Greenplum environment, the benefits of using an appliance has great appeal for scalability, price point, and support. Benefits of implementing the Pivotal DCA include:

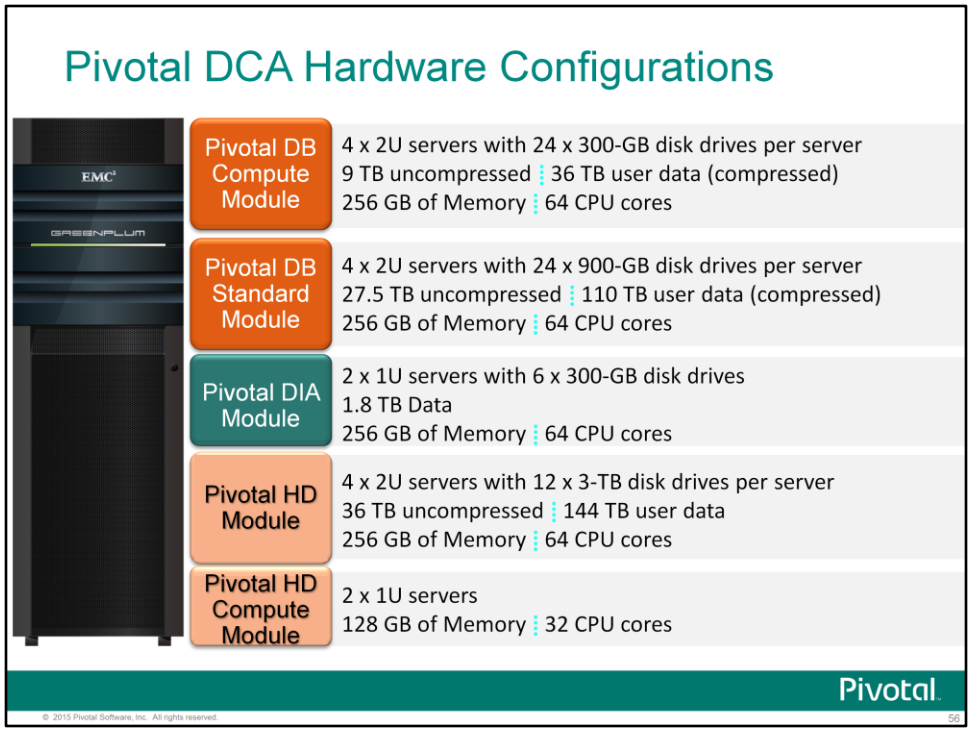
- A seamlessly installed and integrated system that provides lower total cost of ownership (TCO) and higher return on investment (ROI).
- Multiple DCA solutions that provide price and performance leadership for data warehouse hardware solutions
- The industry's first enterprise data cloud platform to support virtualized data warehouses and an analytic infrastructure
- The MPP approach and flexible scalability to meet increasing service level demands make it a price/performance leader in today's market. It supports the Greenplum Database shared-nothing, MPP architecture by moving the processing dramatically closer to the data and users, allowing resources to process every query in a fully parallel manner. All storage connections are used simultaneously while data flow is efficiently managed between resources when performing moves as a result of a query plan.
- Next generation analytic processing lets you quickly analyze Big Data to address the fast-paced environment
- High availability, fault tolerance and the robust analytic processing of large streams of data further support the overall benefits of a data computing appliance approach to data warehousing and business intelligence.



Through its new modular support, the Pivotal DCA supports at a minimum, one Greenplum Database Module, and three optional Greenplum Database modules, to allow you to customize your environment as you need. The Pivotal HD option supports a minimum of a two Pivotal HD module – the first module contains the master servers while the second module provides worker servers.

You can get results faster by using an integrated appliance that offers optimized performance, ease of deployment, increased system monitoring and manageability, and a reduced footprint. The Pivotal DCA modules greatly simplify the expansion of capacity and performance of the Greenplum Database (analytic database) and Pivotal HD (Apache Hadoop) portions of the systems. This data management appliance delivers maximum flexibility and scalability for organizations that are tackling terabyte- to petabyte-scale data opportunities.

All modules utilize the Pivotal DCA interconnect to provide high-speed, high-performance, low-latency connections. This speed is maintained whether a single rack or multiple racks are employed in the cluster.



The Pivotal DCA is a modular solution that lets you incorporate Greenplum Database, Pivotal HD, and third-party solutions in an expandable rack solution. Configurations support ranges from a single Pivotal Database, formerly Greenplum Database, or Pivotal HD module to a 12-rack solution with a mix of modules.

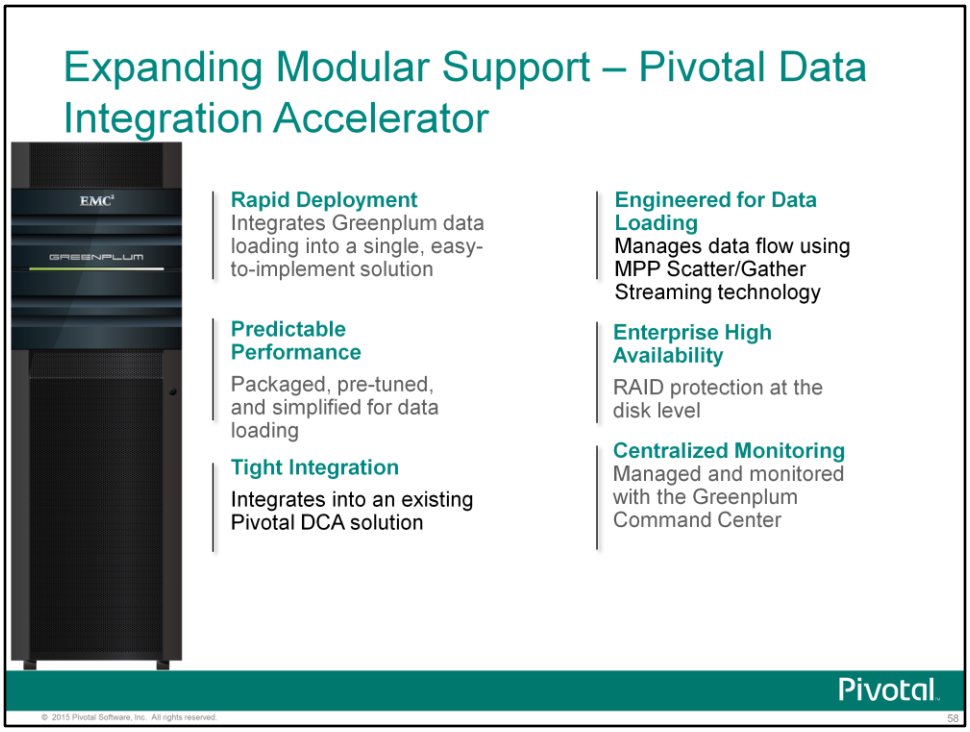
Supported modules include:

- Pivotal Database Compute Module offers is a cost-effective solution for those who require large computational power but less storage needs.
- Pivotal Database Standard Module, which supports the Greenplum Database, and offers the best price to performance ratio with linear scalability. Additional modules can expand the system to accommodate the growing needs of the organization. A full rack offers a scan rate of 40 GB/sec with a data load rate of 16 TB/hour.
- Pivotal Data Integration Accelerator module provides fast integration for partner analytics applications to the Pivotal Data Computing Appliance. With its direct access to the Pivotal DCA 10 gigabit per second interconnect, applications such as Informatica, Talend, and Pentaho can take advantage of the MPP architecture to integrate data into the Greenplum environment.

Pivotal DCA Hardware Configurations (continued)

- Pivotal HD and Pivotal HD Compute modules support unstructured data with Hadoop. The Pivotal HD module supports in-rack storage and computation for Hadoop services, while the Pivotal HD Compute module provides users with the opportunity to extend and scale storage needs with Isilon OneFS scale-out NAS for remote storage.

Each module has CPUs, memory, disk I/O, and interconnects optimally balanced for Greenplum Database, Pivotal HD, and party software for Business Intelligence, ETL, analytics, and data visualization. With the exception of the Pivotal HD Compute module and the DIA which both have two servers, all other modules support four servers each. A cluster can span 1 to 48 modules, yielding 36TB to 5PB of total user data capacity. Total compute capacity ranges from 64 to over 3000+ CPU cores with a multi-rack solution. Various configurations may limit the number and types of modules you incorporate into the final solution.

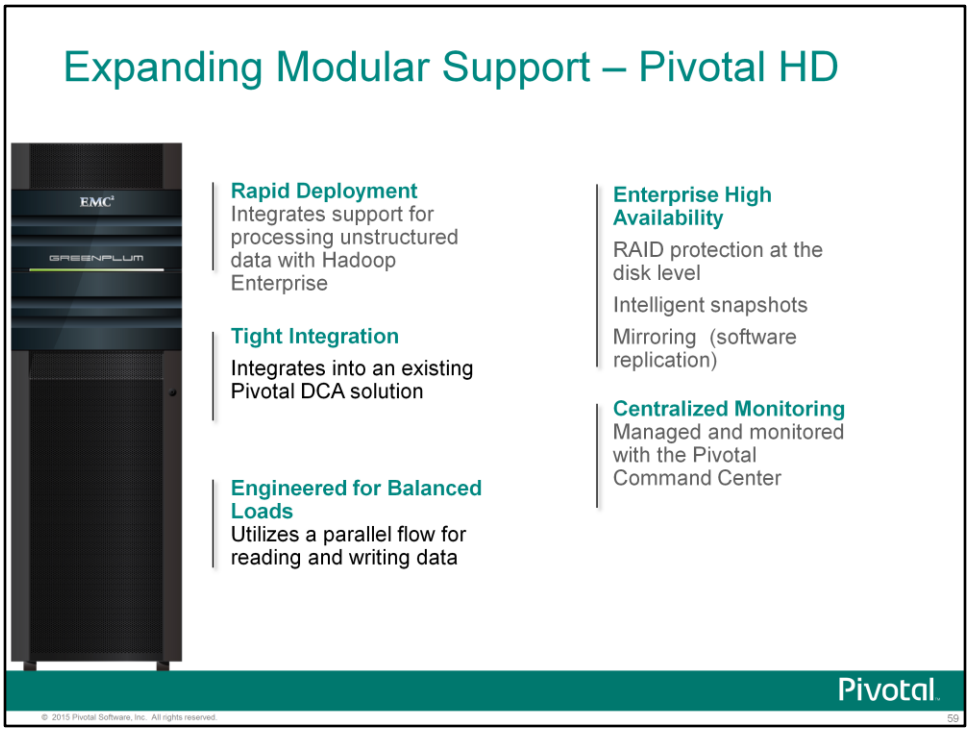


The Pivotal Data Integration Accelerator, otherwise known informally as the ETL module, is a purpose-built, open systems data accelerator that architecturally integrates data loading software, server, storage, and networking into a single, easy-to-implement module. This module is installed directly into the Pivotal Data Computing Appliance rack and provides predictable performance through packaging and pre-tuning for data loading activities.

The module allows implementers to install partner analytics applications and data management applications into the Pivotal DCA. Pivotal applications, such as Greenplum Command Center, Pivotal Command Center, Pivotal Chorus, and Pivotal VRP can be installed in this module, reducing the potential impact on database modules in the rack or the need to use separate servers to support the applications.

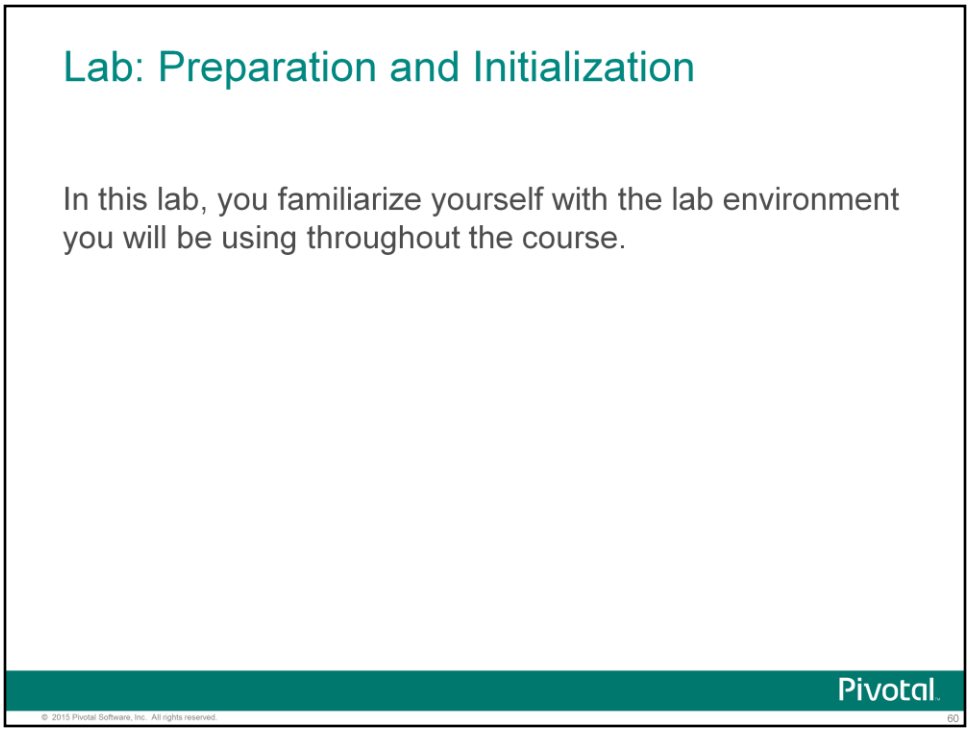
Its main features include:

- Tight integration within a Pivotal DCA to provide an end-to-end solution
- Parallel data loading solution by using the MPP Scatter/Gather Streaming technology to send data from all nodes on the DIA to every segment server of the database
- Disk RAID protection to provide mission-critical support to enterprises
- Configuration support through the Greenplum setup and configuration tool
- Management support through Greenplum Command Center

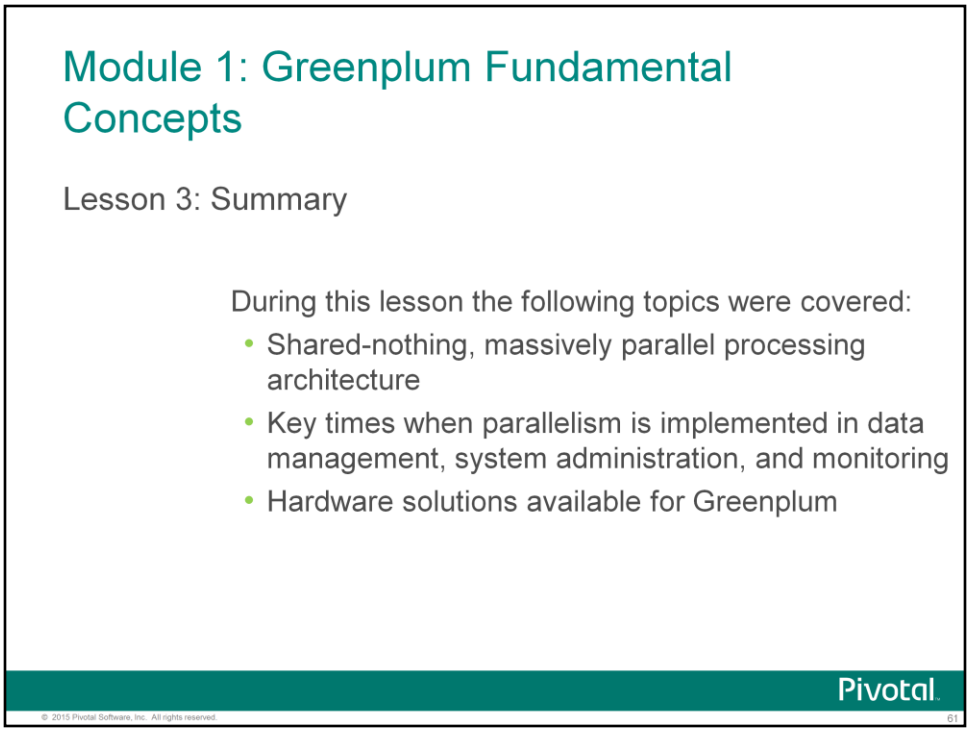


The Pivotal HD modules fuse Hadoop with the Greenplum Database, offering support for co-processing of both structured data with the Greenplum Database, and unstructured data with enterprise-ready Hadoop infrastructure. The module continues to provide enterprise high-availability with RAID protection, intelligent snapshots and mirroring at the software level for the Greenplum Hadoop Enterprise software.

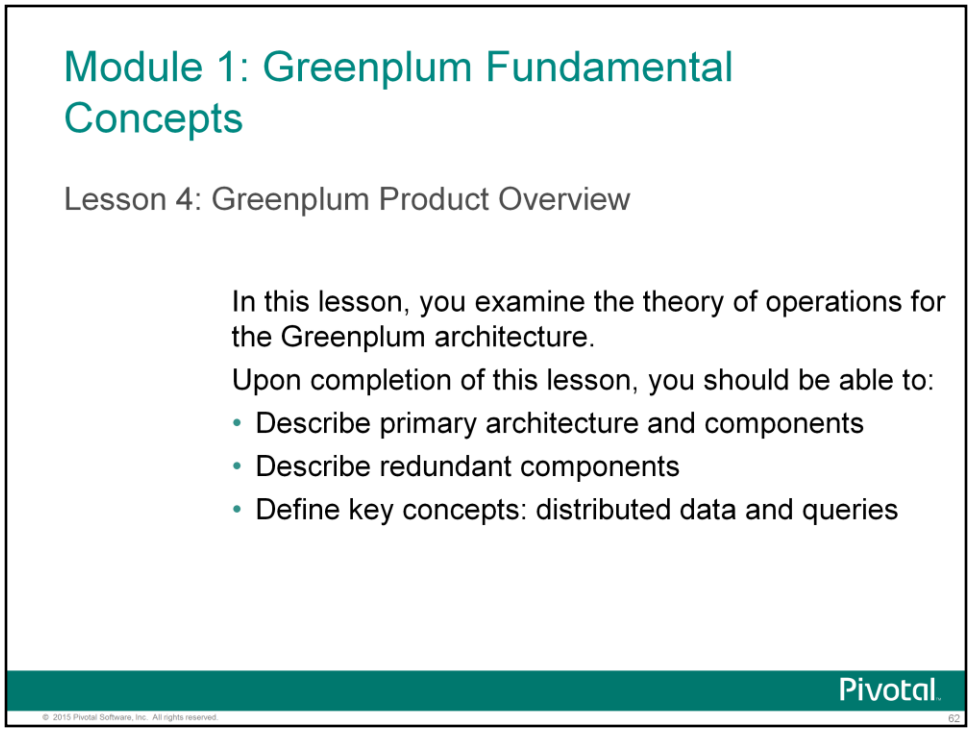
Centralized monitoring offered with Pivotal Command Center. Pivotal Command Center differs from Greenplum Command Center in that it is focused on monitoring and management Pivotal HD modules and Hadoop services.



In this lab, you will access the lab environment and familiarize yourself with the tools you will use throughout the remainder of the course.

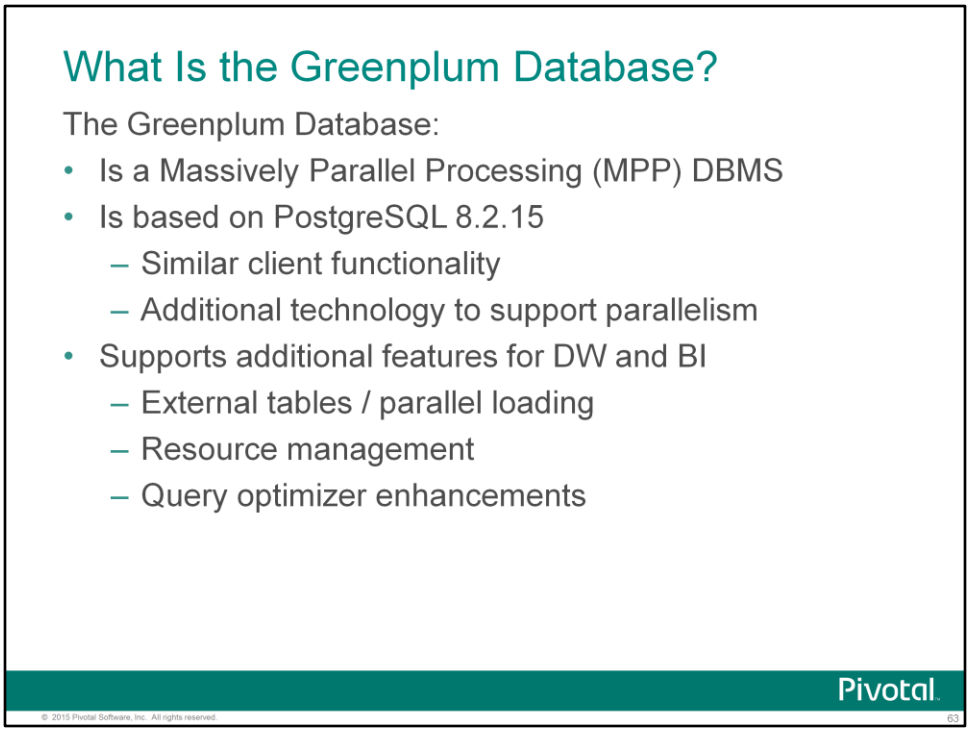


This lesson covered the shared-nothing, massively parallel processing architecture that drives the improved performance and communication found within the Greenplum Database architecture. The lesson also covered the key moments when parallelism is implemented during data management, system administration, and system monitoring of the Greenplum environment, such as during data loads and data scans. Hardware solutions for the Greenplum Database were also presented, specifically the Pivotal Data Computing Appliance and available modules for the appliance.



In this lesson, you:

- Examine the theory of operations for the Greenplum architecture. To do that, you will gain an deeper insight of the components of the architecture introduced earlier and examine how data moves from the client to the segments and back to the client again.
- Identify the components within the architecture that are key to maintaining high availability for the Greenplum environment.
- Define key concepts, including distributed data and queries.



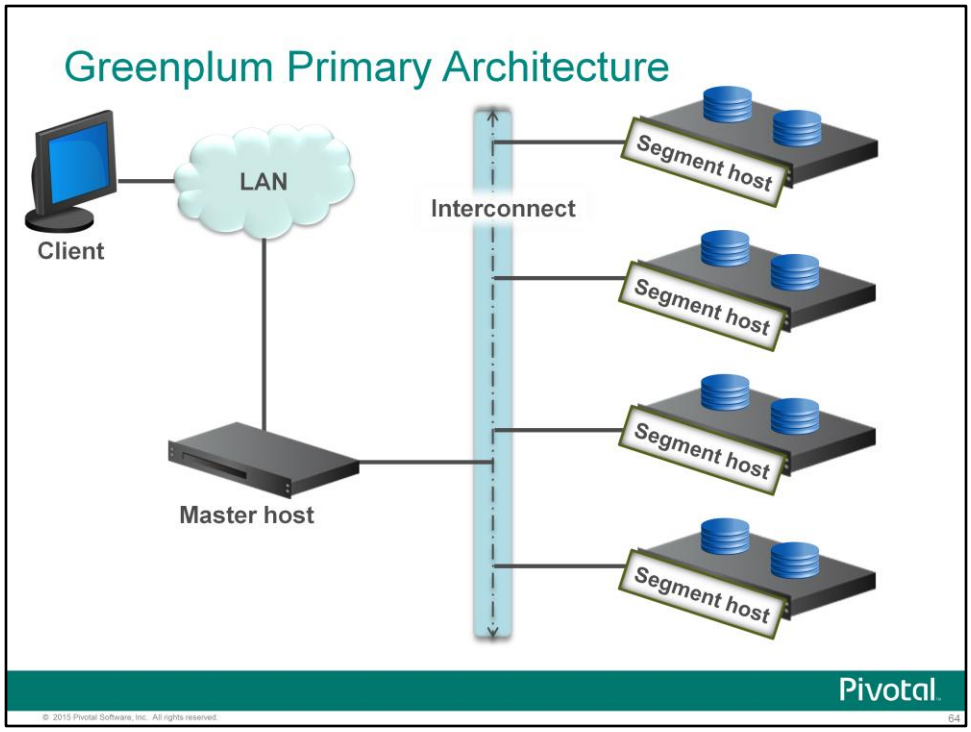
MPP, also known as a *shared-nothing* architecture, refers to systems with two or more units or nodes that cooperate to carry out an operation. Each node has its own memory, operating system, and disks. Greenplum leverages this high-performance system architecture to distribute the load of large volume of data warehouses, and is able to use all of a system's resources in parallel to process a query.

Greenplum Database is:

- Several PostgreSQL database instances acting as one cohesive database management system.
- Based on Postgres 8.2.15 and, in most cases, is very similar to current-day PostgreSQL with regards to SQL support, features, configuration options, and end-user functionality. Database users interact with Greenplum DB as they would a regular PostgreSQL DBMS.

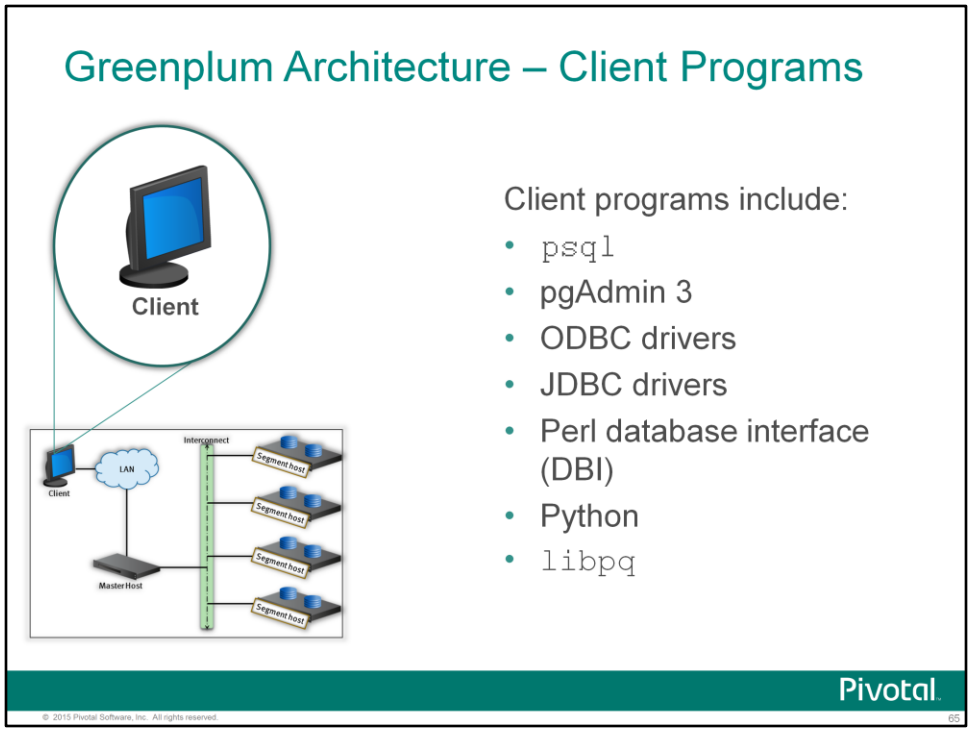
The internals of PostgreSQL have been modified or supplemented to support the parallel structure of Greenplum DB. For example the system catalog, query planner and optimizer, query executor, and transaction manager components have been modified and enhanced to be able to execute queries in parallel across all of the database instances at once. The interconnect component enables communication between the distinct PostgreSQL instances and makes the system behave as one logical database.

Greenplum Database also includes features designed to optimize Postgres for BI and DW such as external tables (parallel data loader), resource management, and query optimizer enhancements. Many BI features developed by Greenplum do make their way back into the PostgreSQL community. For example, table partitioning is a feature developed by Greenplum which is now in standard Postgres.



This slide introduces each primary or required component of a Greenplum DB system at a high-level. We will cover each component in detail on the next few slides.

- Users access a Greenplum database using various client applications and interfaces. All client interfaces supported by PostgreSQL are also supported by Greenplum Database.
- The access point to a Greenplum Database system is the master – clients always connect to the master instance. The master instance is the database process that accepts client connections and processes the SQL commands issued by the users of the system. The master does not contain any user data, storing only the system catalogs or metadata.
- The master coordinates the work among all of the segment instances in the system. Data resides in the segments. This is where the majority of the query processing takes place. User-defined tables and indexes are distributed across the available number of segments in the Greenplum system, each segment containing a distinct portion of the data.
- The Interconnect is the glue that brings all the distinct database instances together to act as one cohesive DBMS. The Interconnect component in Greenplum Database is responsible for moving data between the segments during query execution. The interconnect delivers messages, moves data, collects results, and coordinates work among the master and segments in the system. The Interconnect rides on top of a standard Gigabit Ethernet switching fabric over a (preferably) private local area network (LAN).

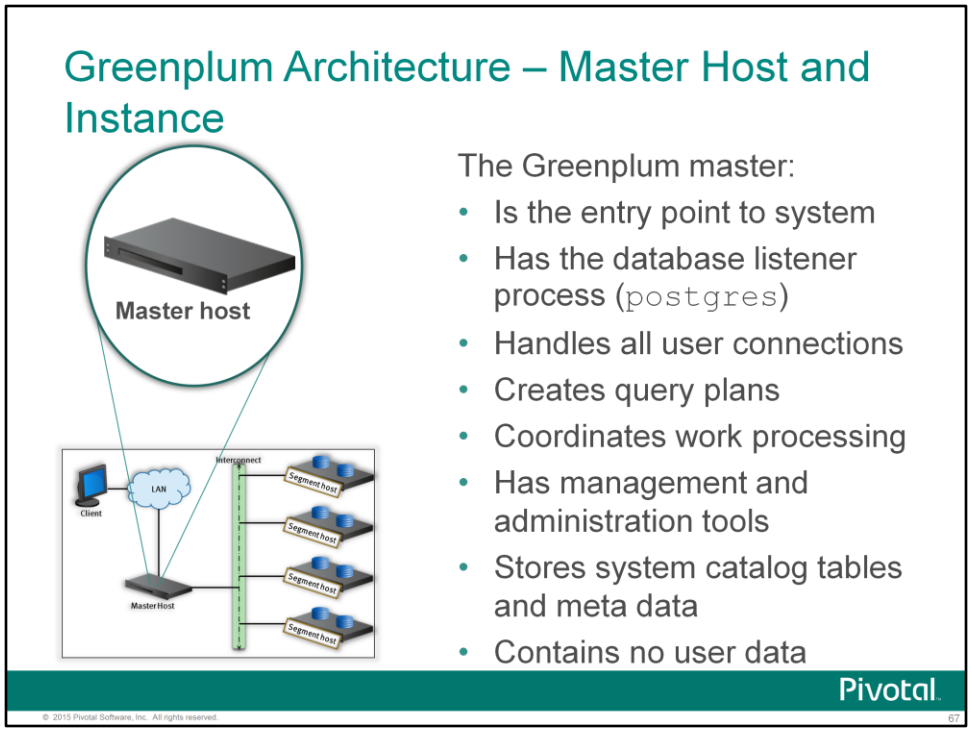


Greenplum Database uses the same client interfaces as PostgreSQL. Although there are other client interfaces available, the following client interfaces have been tested with Greenplum Database:

- **`psql`** – The command-line terminal interface to PostgreSQL, psql is bundled with your Greenplum installation. This terminal-based front-end command to PostgreSQL lets you access the PostgreSQL database interactively or through a file. It also provides meta-commands and other shell-like features to facilitate writing scripts and automating a variety of tasks.
- **pgAdmin 3** – pgAdmin is a graphical interface developed by an international team of PostgreSQL developers. It is a feature-rich, open-source administration and development platform. It contains a graphical interface to support all PostgreSQL features. It is free and available for download on many platforms.
- **ODBC** – Using the Postgres ODBC database driver, `psqlodbc`, Open Database Connectivity-compliant client applications, such as ETL or reporting tools, can be configured to connect to a Greenplum database.
- **JDBC** – The Postgres Java Database Connectivity (JDBC) database driver, `pgjdbc`, allows Java programs to connect to a PostgreSQL database using standard, database independent Java code. It provides reasonably complete implementation of the JDBC 3 specification in addition to some PostgreSQL specific extensions.

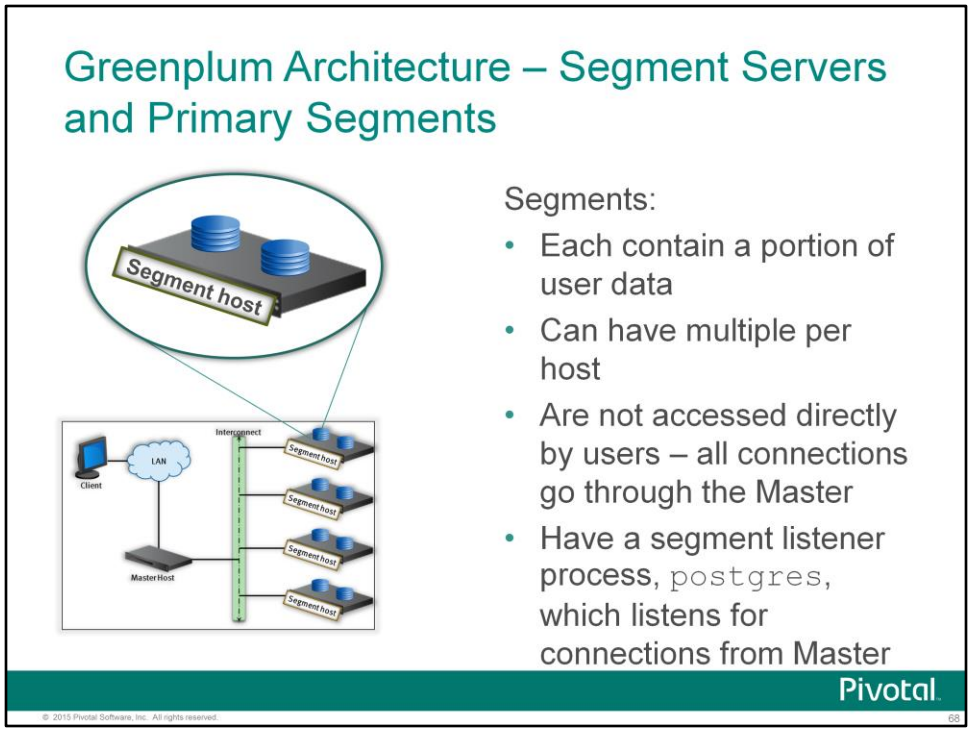
Greenplum Architecture – Client Programs (continued)

- **PERL DBI** – PERL Database Interface is an API for connecting programs written in Perl to database management systems (DBMS). It is the most common database interface for the Perl programming language. The PostgreSQL Perl database driver, `pgperl`, can be used to access a Greenplum database.
- **Python** – There are several Python interfaces available for PostgreSQL. PyGreSQL is the one recommended on the postgresql.org web site.
- **`libpq`** – The PostgreSQL native C application programming interface to PostgreSQL, `libpq`, is a set of library functions that allow client programs to pass queries to a PostgreSQL backend server and to receive the results of these queries. `libpq` libraries are distributed with Greenplum Database.



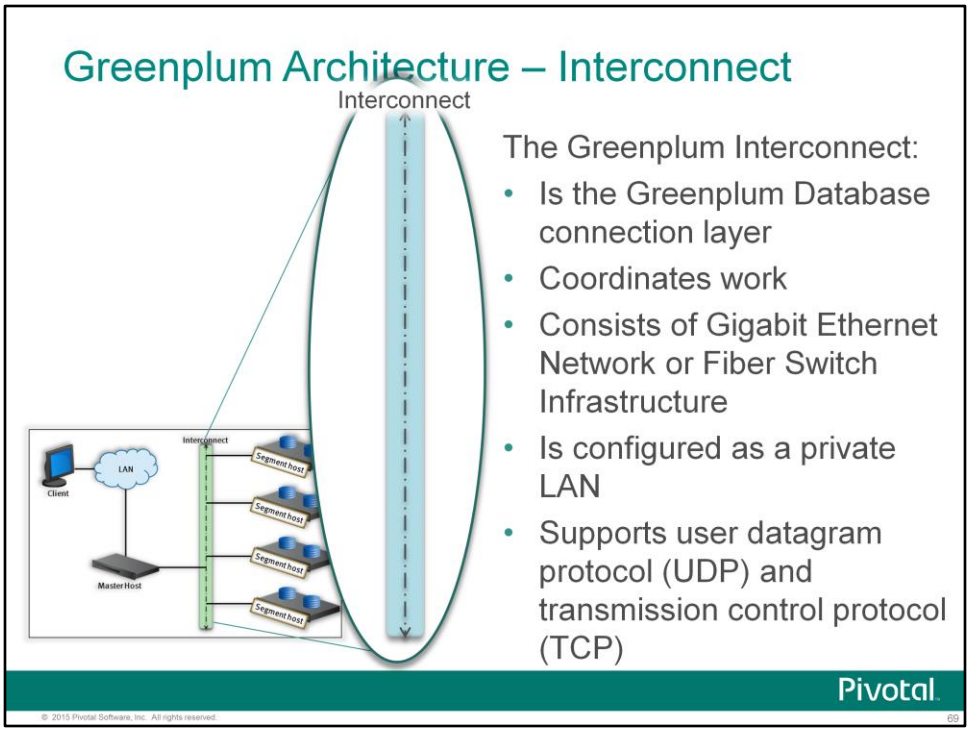
The Greenplum Database Master instance or master:

- Is the entry point to a Greenplum Database system. Users connect to the master and interact with a database as they would in any other DBMS. Any SQL statement or data load goes through the master, always.
- Has the PostgreSQL database listener process, `postgres`, also referred to as the *postmaster* in prior releases. This is the process to which users connect and initiate a database session. By default, it uses port 5432.
- Handles connections, authenticates users, and processes incoming requests.
- Creates query plan to distribute the query workload across all segment instances.
- Occasionally does some final processing for queries such as final aggregations, summations, orders and sorts. However, most of the workload of a query is handled on the segment instances.
- Is the entry point for all system administration tasks. All of the administration utilities and tools for Greenplum Database reside on the master installation.
- Does not contain any user data. User data is distributed on the segment instances. The only data contained on the master is the system catalog tables and system metadata.



Segment servers and their segments have the following characteristics:

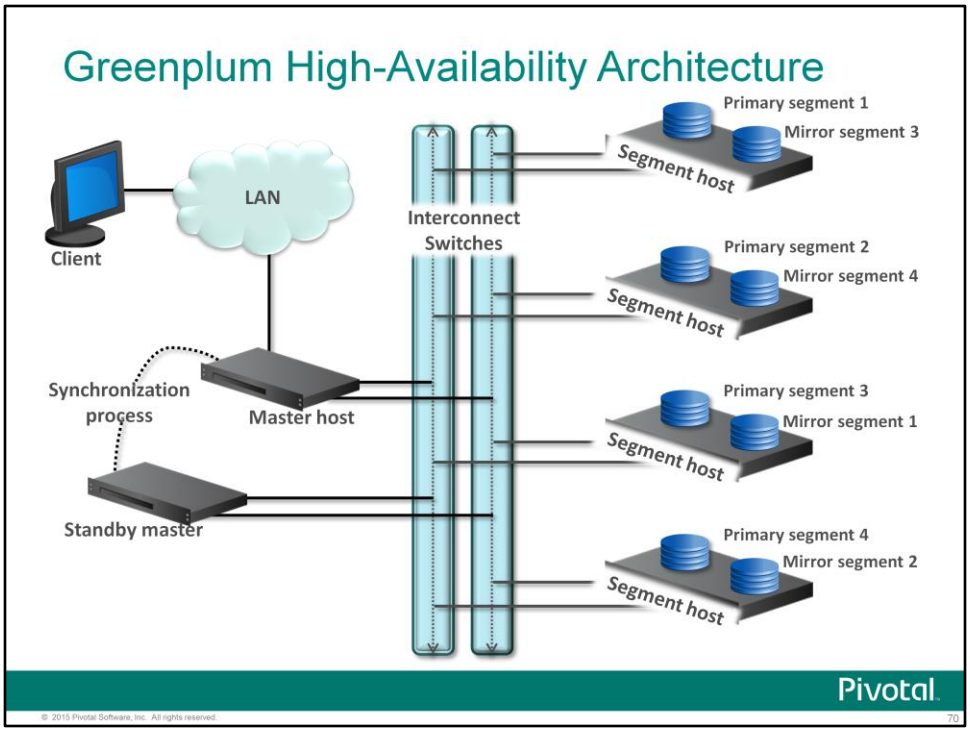
- All user-created tables and indexes are distributed across the segment instances or segments in the system. Each segment holds a portion of data for each distributed user table and index.
- A segment host can have multiple segment instances residing on it. Typically there is one primary segment instance per CPU core on a host in a production installation. If mirroring is enabled, there is a primary and mirror pair per CPU core.
- Users and administrators do not access the segments directly. All communications with the segments are through the master. While there is a way to connect to a segment in utility mode, the case for doing that is very rare and should not be done without talking to Greenplum support first.
- Each segment instance has a PostgreSQL segment listener process, `postgres`, that listens for connections from the master only. It will not accept client connections from database users. The segment port numbers are configurable at the time the system is initialized.



The Interconnect component in Greenplum Database:

- Is the connection layer between all of the individual database instances, master and segments. It is the glue that holds all of the components together and makes the system act as one cohesive DBMS. This is the proprietary component developed by Greenplum that enables parallel processing.
- Is responsible for moving data between the segments during query execution. The interconnect delivers messages, moves data, collects results, and coordinates work among the segments in the system.
- Supports gigabit Ethernet network or fiber switch infrastructure as configured in the Pivotal Data Computing Appliance cluster.
- Is configured as a private LAN. Segment hosts should not be visible outside the Greenplum array.
- By default, the Interconnect uses user datagram protocol (UDP) to send messages over the network. The Greenplum software performs the additional packet verification and checking not performed by UDP, so reliability is equivalent to transmission control protocol (TCP). UDP performance is equivalent to or better than TCP.

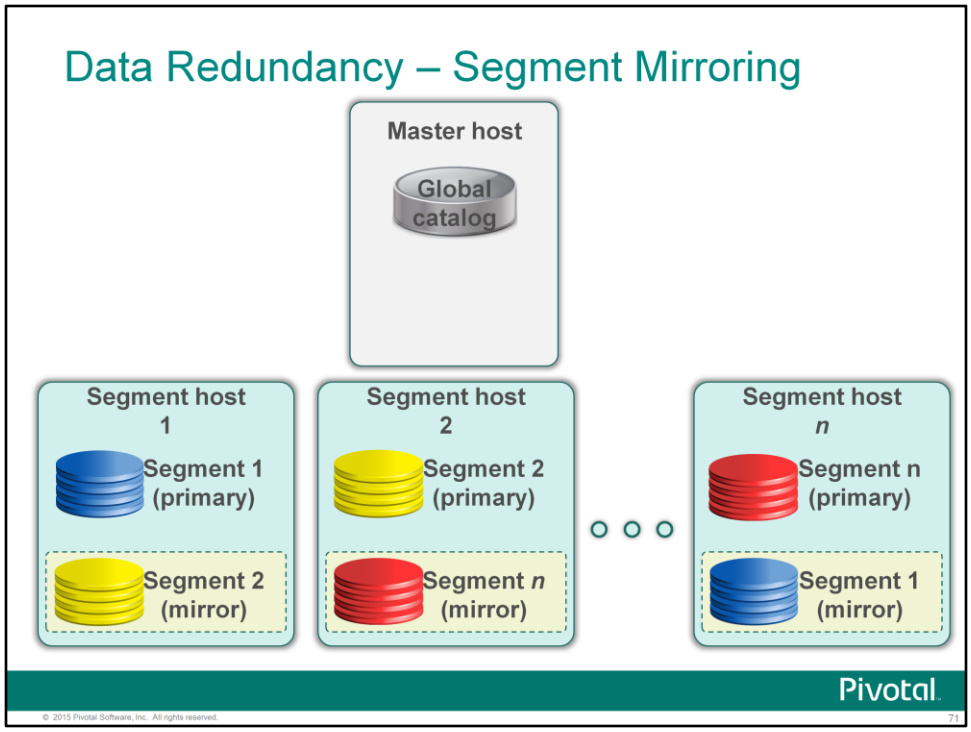
Note: If performing data loads with ETL, the ETL server can be plugged directly to the interconnect so that it can communicate with the segment servers directly.



In addition to the primary Greenplum system components, you can also optionally deploy redundant components for high-availability and no single-point of failure.

For data redundancy, you can implement the following configuration:

- **Mirror segment instances** – A mirror segment always resides on a different host than its primary. Mirroring provides you with a replica of the database contained on a segment which may be useful if you lose a host in the array or a segment instance becomes corrupted, as may occur with disk failure.
- **Standby master host** – For a fully redundant Greenplum Database system, a mirror of the Greenplum master instance can be deployed. A backup Greenplum master host serves as a *warm standby* in the event that the primary Greenplum master host becomes inaccessible. The standby master host is kept up to date by a transaction log replication process that runs on the backup master host and keeps the data between the primary and standby master synchronized. Should the master host fail, log replication is shut down, the standby master is activated, and the transaction logs are used to reconstruct the state of the master at the time of the last successful transmission.
- **Dual interconnect switches** – A highly available Interconnect can be achieved by deploying dual Gigabit Ethernet or Fiber switches and redundant network interfaces on all the Greenplum Database hosts in the system. The default configuration is to have one network interface per primary segment instance on a segment host, each configured on its own subnet. If using dual switches, divide the subnets evenly between the switches.



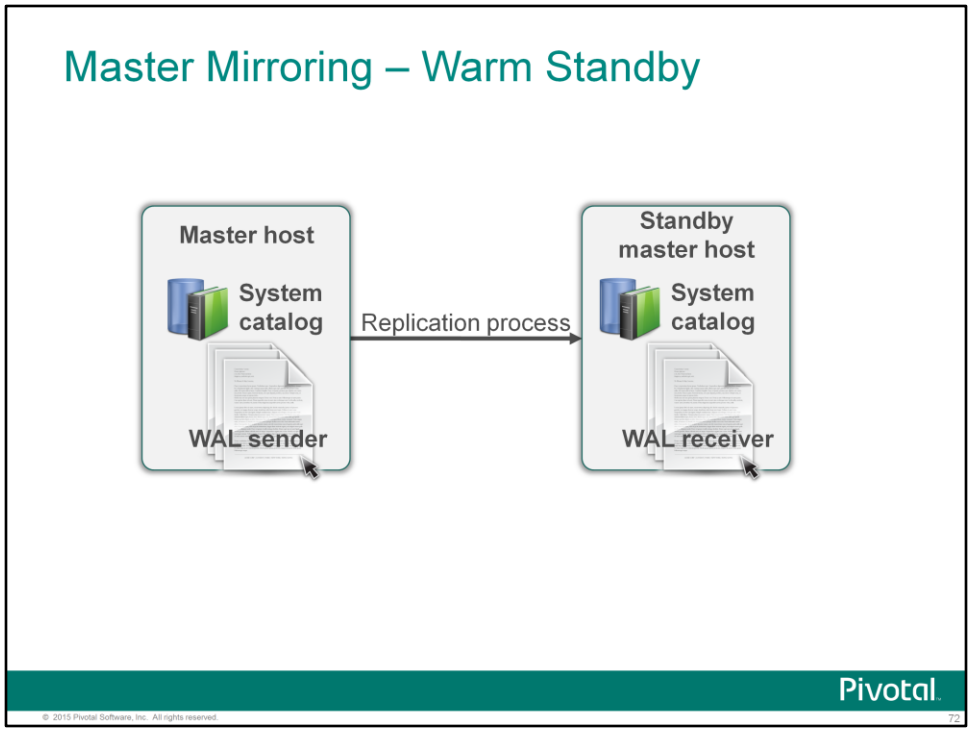
In a Greenplum system, the master host stores only the system metadata. Each segment host has a distinct portion of the user data. With mirroring, a replica of a segment database resides on a different host. The mirror acts as an exact replica of the primary segment and is only activated should the primary segment fail. The mirror segment receives 32k pages of changes made to the primary.

The following is an example of a redundant configuration:

- If the primary copy for Segment 1 is on segment host 1, the mirror copy is on another segment host, segment host *n*.
- The primary copy for Segment 2 is on segment host 2. Its mirror copy is on segment host 1.
- The primary copy for segment *n* is on segment host *n*. Its mirror copy is on segment host 2.

In this scenario, should you lose a segment host, you can continue to access all of the data segments as queries will fail over to the mirror segment should the primary segment be unavailable. Spread mirroring helps to balance the node so that should a segment fail, all of the remaining segments can distribute the load.

While the default mirror setup uses the same hosts as where your primaries are deployed, note that it is also possible to deploy your mirrors on a completely different set of hosts than your primaries, even hosts in a different physical location.

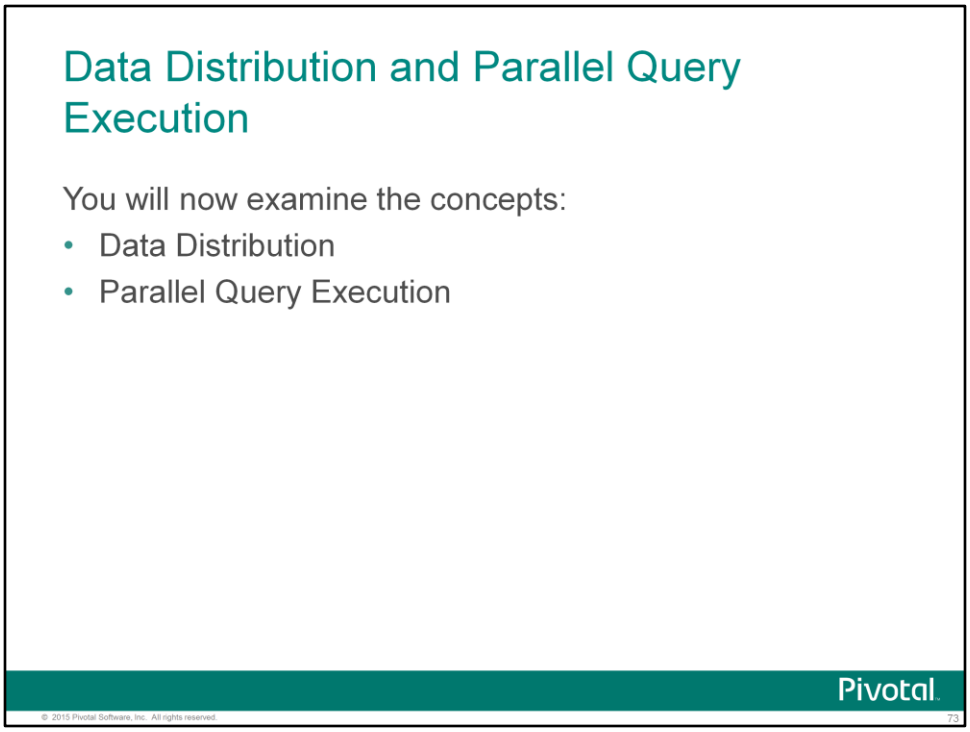


Because the Master is the entry point to a Greenplum Database system, you may want to have a backup in case your primary master hosts becomes unavailable. You can do this by deploying a *warm standby master*. The standby master host is kept up to date by replication processes executing on both the master and standby servers. The `walsender` process executes on the master and streams committed transactions to the `walreceiver` on the standby server. This streaming replication is used to keep the standby master synchronized with the master server.

If the primary master fails, the log replication process is shutdown, and the backup master can be activated in its place.

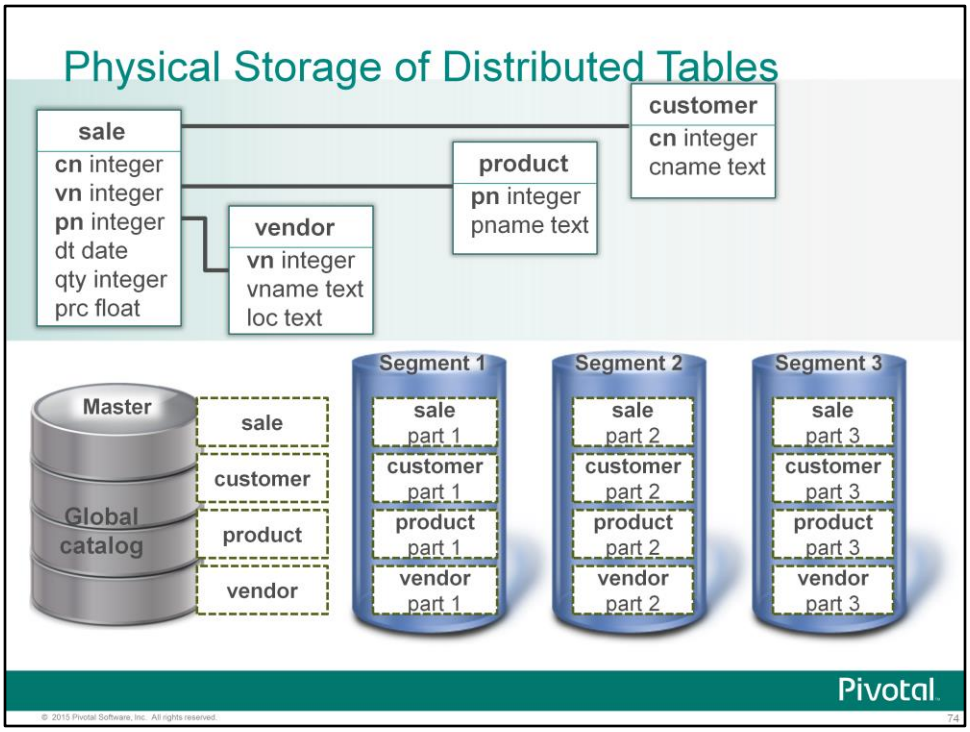
Upon activation of the backup master, the replicated logs are used to reconstruct the state of the Greenplum master host at the time of the last successfully committed transaction.

Since the Greenplum master does not contain any user data, only the system catalog tables need to be synchronized between the primary and backup copies. These tables are not updated frequently, but when they are, changes are automatically copied over to the backup Greenplum master so that it is always kept current with the primary.



In the next set of slides, you will examine key Greenplum Database concepts, including data distribution and parallel query execution.

Parallel query execution A lot of systems available in the marketplace today have parallel query execution. However, those databases do not support distributed tables Greenplum is a true MPP system in that this is a shared-nothing architecture.



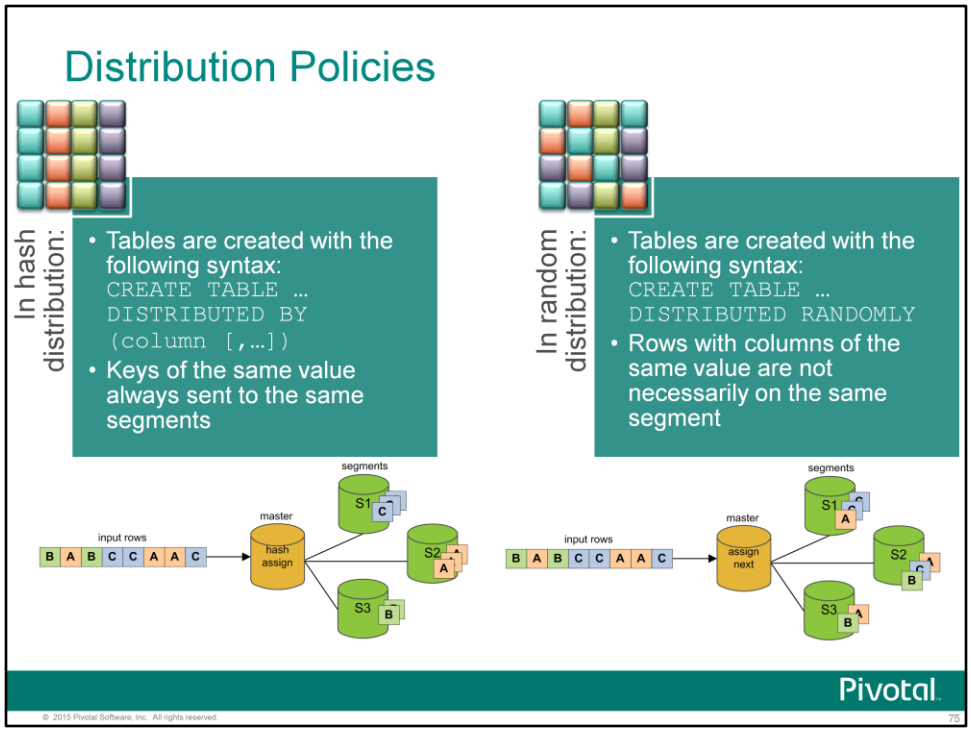
To understand how Greenplum Database stores data across the various hosts and segment instances, consider the following simple logical database schema.

This shows a simple star schema common in data warehousing. In this type of database schema, the sale table is usually called a fact table and the other tables (customer, vendor, product) are usually called the dimension tables.

Now let's see what these tables look like in the physical database.

In Greenplum Database, all tables are distributed. This means a table is divided into non-overlapping sets of rows or parts. Each part resides on a single database known as a segment within the Greenplum Database system. The parts are distributed evenly across all of the available segments using a sophisticated hashing algorithm.

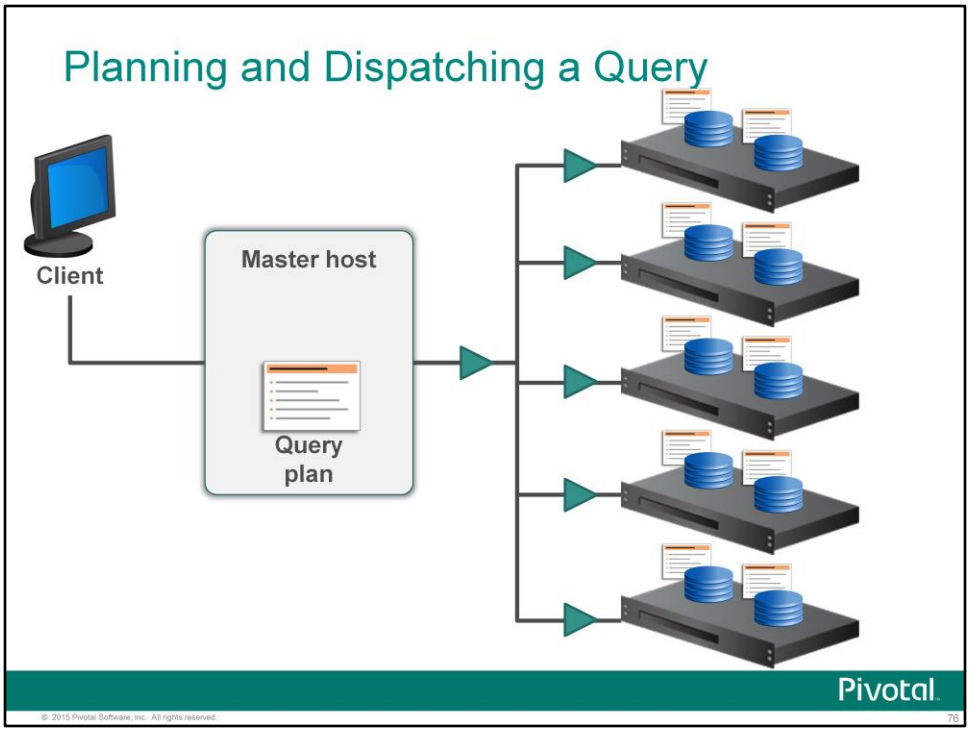
Distribution is determined at table create time by selecting a distribution key of one or more columns. Typically you would use a table's primary key or some other unique column or set of columns as the distribution key.



There are two distribution policies for dividing the rows among the available segments:

- **Hash** – In hash distribution, one or more table columns are used as the distribution key. These columns are used by a hashing algorithm to divide the data among all of the segments. The key value is hashed, or a random number created. This hashed value is divided by the number of segments. The remainder of this division is the assigned segment number. Keys of the same value will always hash to the same segment. Choosing a unique distribution key, such as the table's primary key, will ensure the most even data distribution.
  Note: There are performance advantages to choosing a hash policy whenever possible. We will examine the performance implications of choosing one over the other later in the course.
- **Random** – In random distribution, a random number is used as the key to generate the hash. Once generated, the same behavior used for the hash distribution is also used here. The hashed value is divided by the number of segments and the mod is used to assign the row to the assigned segment number. While rows with the same value will not necessarily be assigned to the same segment, the assignment is more random but the distribution is within a 10% variance.

Note: External tables do not have a distribution policy associated with them as they are outside the realm of control for the Greenplum DB.



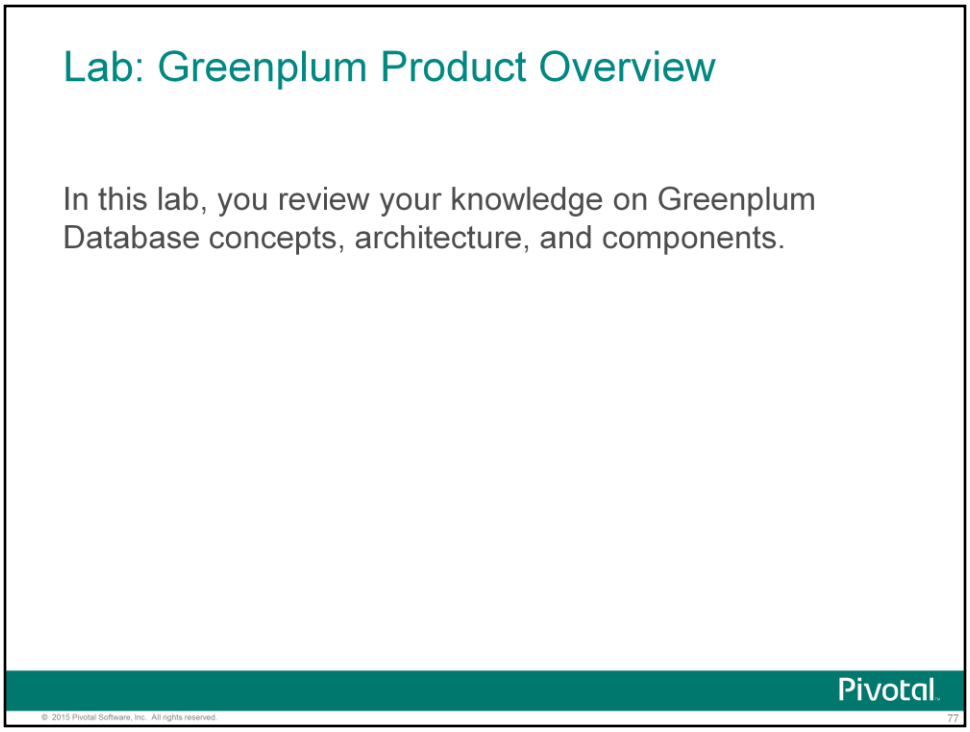
When working with Greenplum, you issue queries the database as you would any other database management system (DBMS). You can connect to the database instance on the Greenplum master host using a client application, such as psql, and submit an SQL statement.

The master:

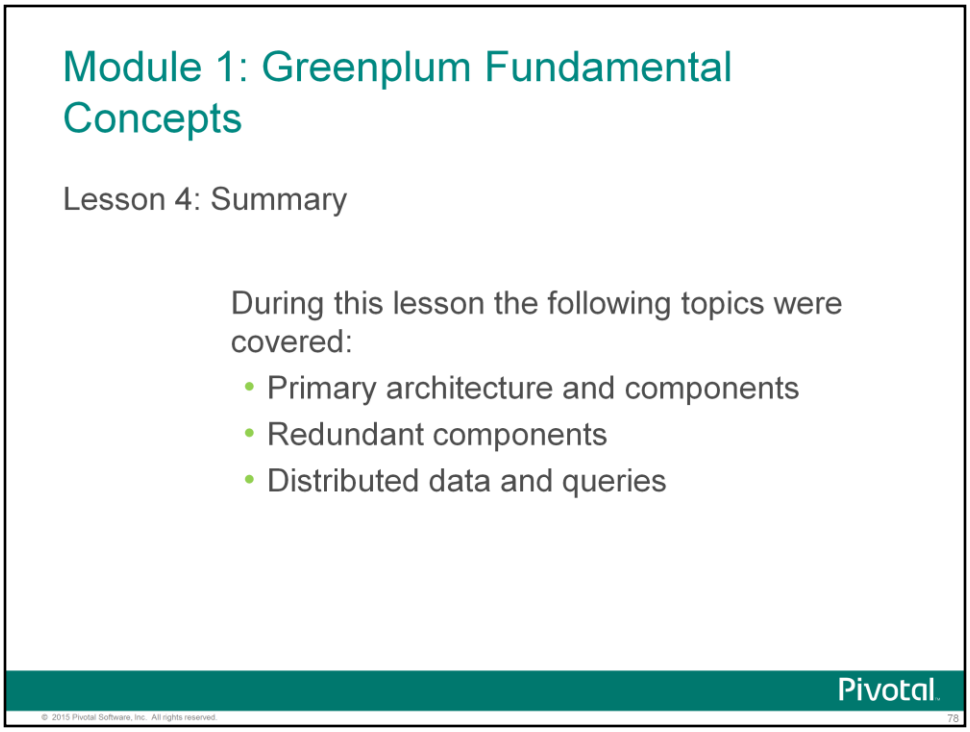
- Receives the query
- Parses the query
- Optimizes the query
- Creates a parallel query plan
- Dispatches the same plan to all of the segments for execution

Each segment is responsible for executing local database operations on its own set of data.

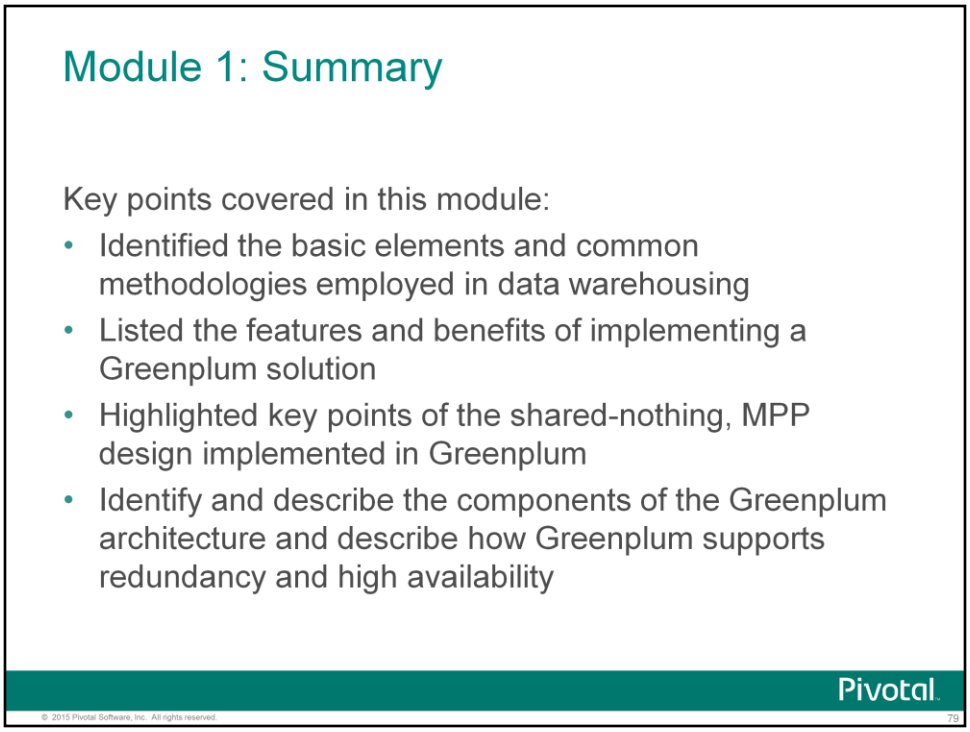
We will examine this concept in great detail later in the course.



In this lab, you will review concepts you have learned in this lesson.



This lesson covered the primary architecture in greater depths, discussing the components found within the Greenplum environment, including the master, standby, and segment hosts, the redundant architecture to safeguard the environment and improve uptime, and how data is distributed and queries managed.



Listed are the key points covered in this module. You should have learned to:

- Define data warehouse, Big Data, and identify the basic elements and methodologies employed in data warehousing.
- List the features and benefits of implementing the Greenplum Database with applicable hardware solutions.
- Identify the key points of the shared-nothing, MPP design in Greenplum.
- Identify and describe the components of the Greenplum architecture and describe how Greenplum supports redundancy and high availability.

This slide is intentionally left blank.

Pivotal