PENTAHO BUSINESS ANALYTICS TRAINING

Rahul Gurjar

Course Outline

- Overview of the Pentaho Stack
- Introduction to OLAP, DWH and Dimension Modelling
- Pentaho BA Server
- Pentaho ETL
- OLAP Schema creation in Pentaho in detail
- Dynamic Security in Pentaho OLAP
- MDX in action
- Pentaho Reporting in detail (PRD and PME)
- Analysis Report
- Dashboard (EE and CDE)
- Introduction to Pentaho Big Data Ecosystem

Training Process

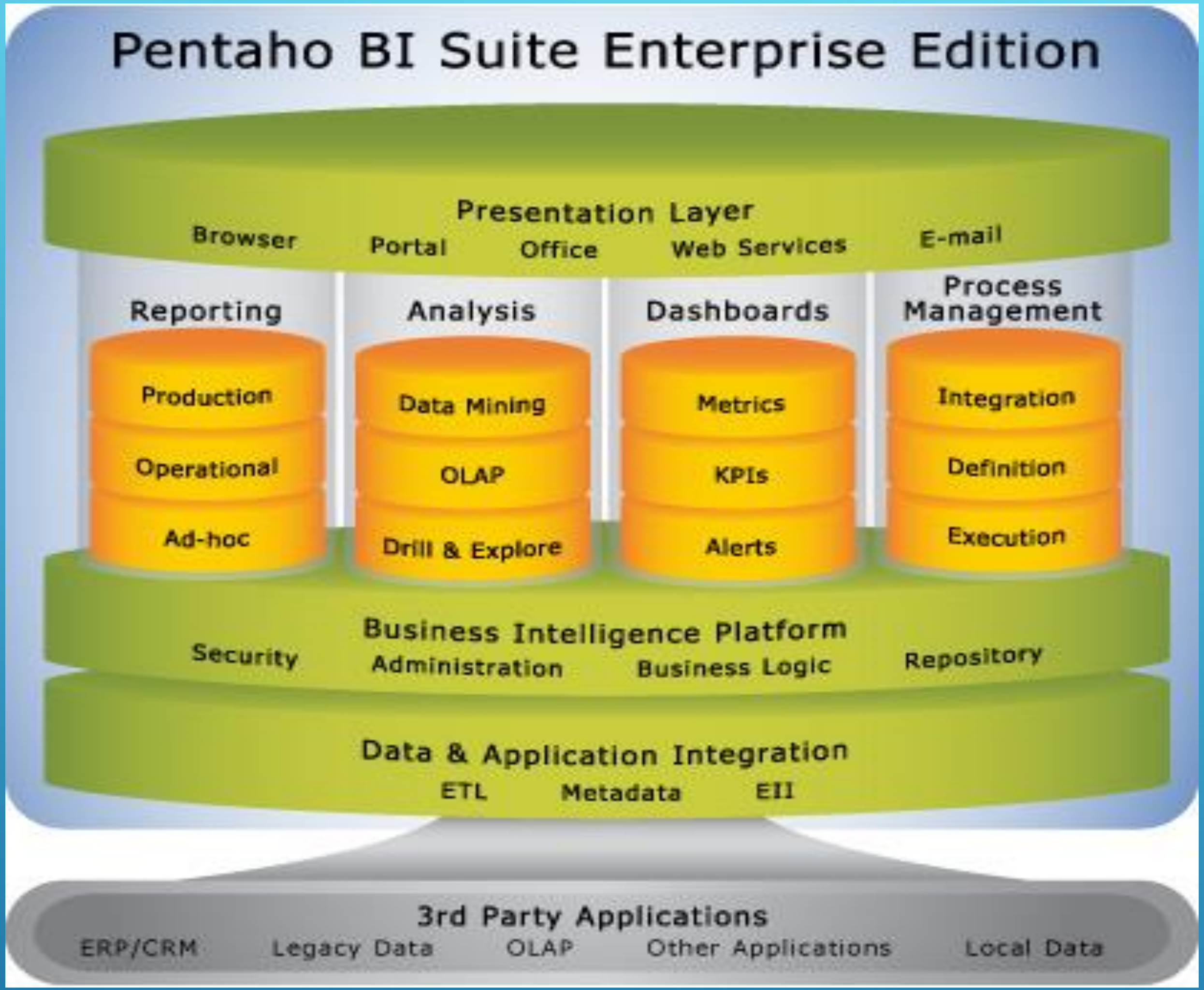
Daily schedule:

- 10:00 am – 6:00 pm
- 1 hour lunch break
- Other breaks as needed

PENTAHO STACK OVERVIEW

PENTAHO BA SERVER COMPONENTS

- Pentaho Architecture Overview
- Pentaho BA Server
- Pentaho Data Integration
- Pentaho Schema Workbench
- Pentaho Report Designer
- Pentaho Metadata Editor



PENTAHO ARCHITECTURE

PENTAHO BA SERVER FEATURES

- User Management
- Helps in Access Control
- Manage Content
- Execute/View Reports
- Manage Data Sources
- Manages Schedules
- Facilitates Self-Service Analysis/Reporting/Dashboard creation

PENTAHO DI FEATURES

- Tool to create and execute ETL flows
- Can connect to a lot of Source and Target Systems
- Metadata driven ETL engine
- The ETLs are simple XML files
- Plugin Based architecture
- Easy to develop and deploy ETL flows

PENTAHO REPORT DESIGNER

- Can connect to various Data Sources including Mondrian
- Seamless integration with the BI Server
- Can ingest data from ETL flows as well
- Developed from the Jfree project
- Can create pixel perfect reports

PENTAHO METADATA EDITOR

- Creates Metadata layer on DB Schema
- Can connect to any database (JDBC)
- Metadata Model for Access Control
- Metadata Model Integrates with BA Server
- MQL is used for data retrieval

PENTAHO SCHEMA WORKBENCH

- Used to develop OLAP Schema
- Mondrian uses these schema to Analysis
- Supports Star and Snowflake schema
- Role based access control

COMMUNITY DASHBOARD EDITOR

- Creates rich Dashboards for the Pentaho BA Server
- Integrated with the BA Server
- Very flexible for the developers
- Connects to various Data Sources including PDI sources

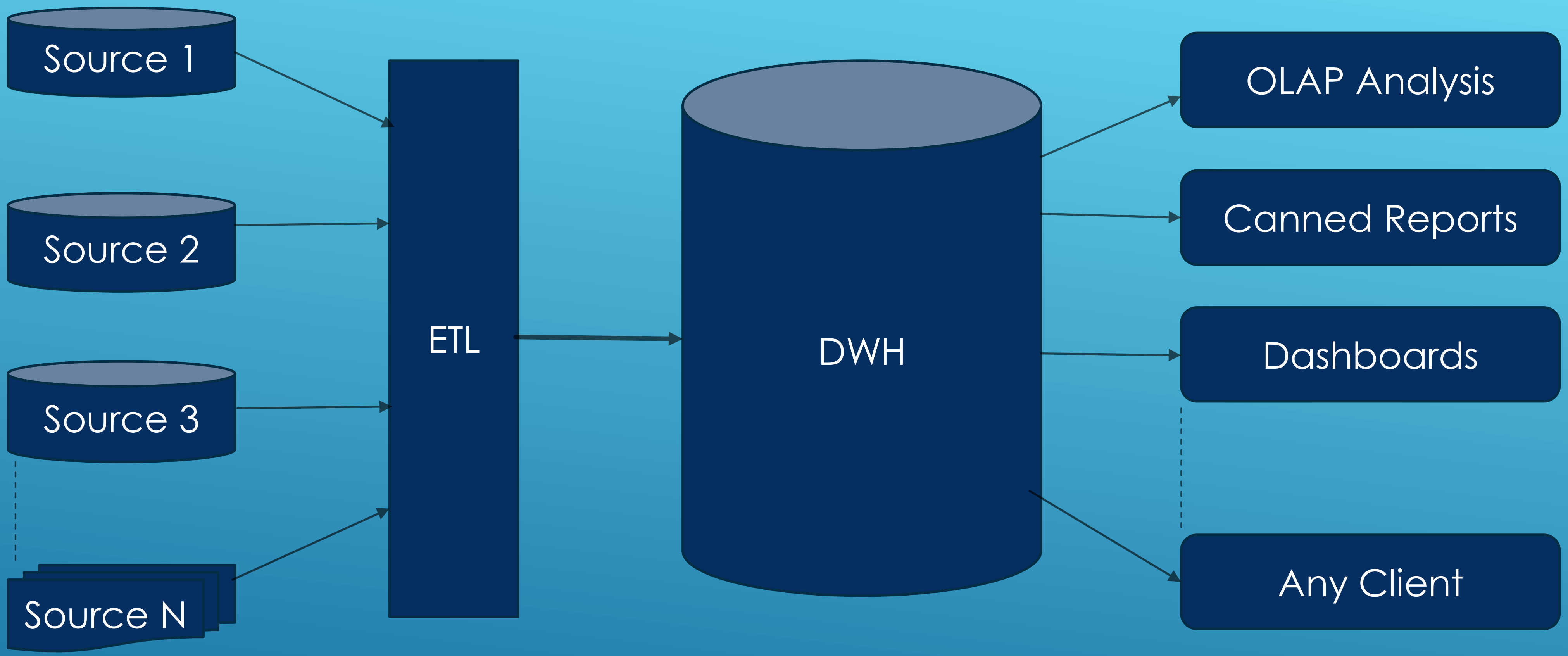
INTRODUCTION TO OLAP, DWH AND DIMENSION MODELLING

OLAP

- OLAP - Online Analytical Processing
- Provides Multidimensional Analysis of Business Data
- Mondrian is the Engine used in the Pentaho Stack
- MDX is the language that talks to Mondrian
- Pivoting can be done using client tools like Analyzer
- Operations supported : Roll up, Drill Down, Slice and Dice.

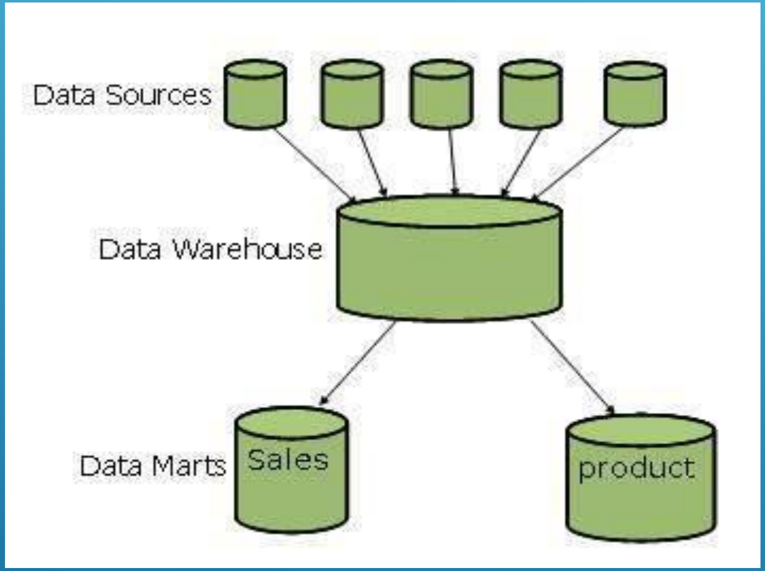
DWH

- Subject Oriented
- Time Variant
- Non Volatile
- Main Data Storage system in the BI world
- Mainly used for Reporting and Analysis
- Separate from Transaction Database
- Central repository for Integrated data from various sources


Source 1
Source 2
Source 3
Source N
ETL
DWH
OLAP Analysis
Canned Reports
Dashboards
Any Client

DATA MART

- Subset of the Organization-wide Data
- Subject Specific
- Smaller in size



DIMENSION MODELING

- Technique and concept used for creating DWH
- Different from ER Modeling
- Facts are Measures
- Dimension are Context that defines facts
- Dimensions have group of hierarchies
- Shared Dimension (Conformed)
- Degenerated Dimension
- Additive, Semi-Additive and Non-Additive Facts
- Factless Fact

PENTAHO BA SERVER

- Tomcat server based Web Application
- User Console:
  - Solution access
    - Solution Navigation
    - Subscribing to an existing schedule
    - Private Schedule creation (access based)
    - Self Service Report Creation (access based)
  - BA Server Administration
    - User and Role Management
    - System Setting
    - Access Control
    - Public Schedule creation
    - Repository and other Cache Refresh
    - Many more…

PENTAHO DATA INTEGRATION

PDI INTRODUCTION

- KETTLE is open source core of PDI Enterprise Edition.
- PDI Enterprise Edition is production-ready
- PDI Features:
  - Versioning
  - Scheduling
  - Real-Time Monitoring
  - Access Control

WHY PDI?

Ease of use:

- 100% metadata driven (define **WHAT** you want to do, not **HOW** to it)
- No extra code generation means lower complexity
- Simple setup, intuitive graphical designers, and easy

Flexibility:

- Never forces a certain path on the user
- Pluggable architecture for extending functionality

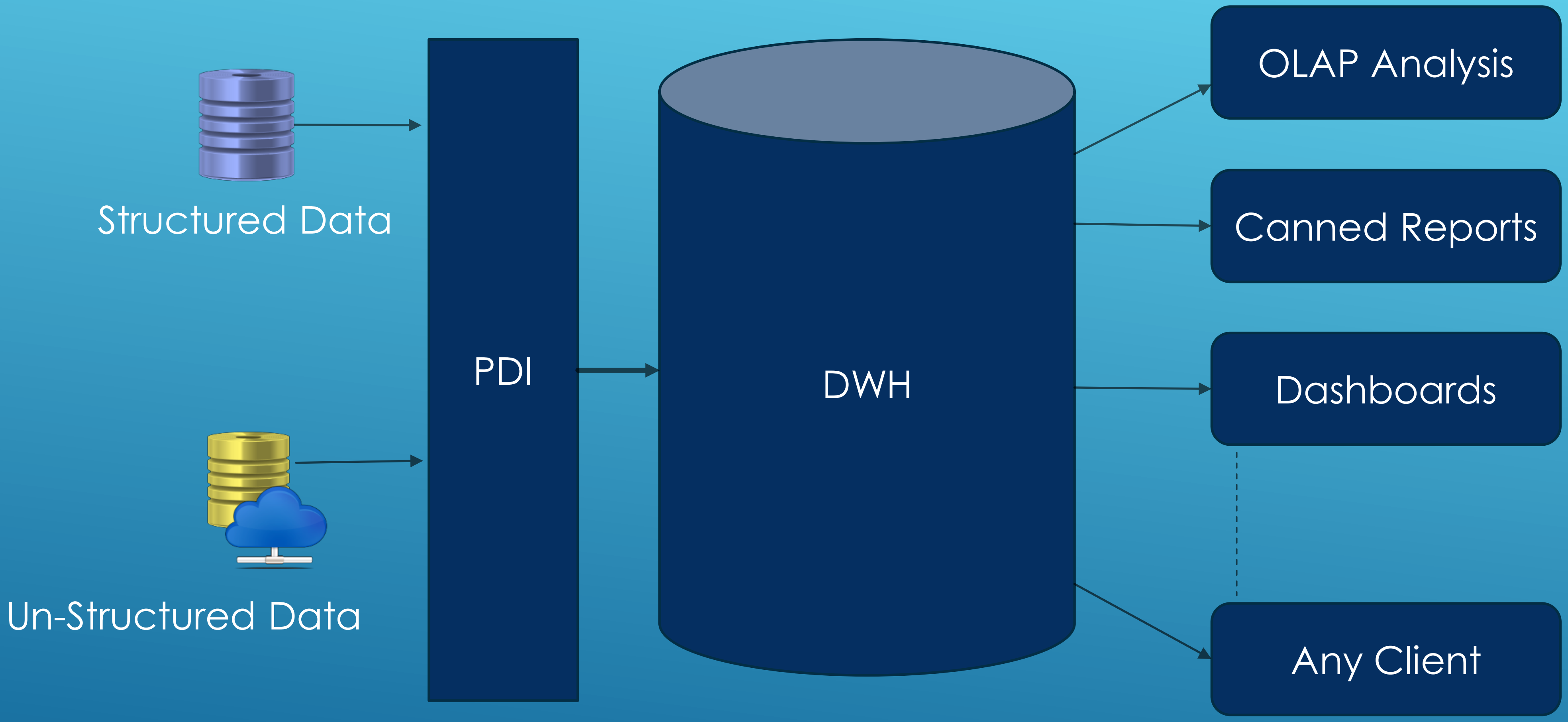
Modern standards-based architecture

- 100% Java with broad, cross-platform support
- Over 100 out-of-the-box mapping objects (steps and job entries)
- Enterprise class performance and scalability

Lower total cost of ownership (TCO)

- Lesser license fees
- Short implementation cycles
- Reduced maintenance costs

WHERE PDI FITS



REPOSITORIES

- Metadata Store for PDI:
  - XML files
  - RDBMS
  - Enterprise
- Objects Stored:
  - Connections
  - Transformation
  - Job
  - Schema
- Components:
  - Spoon
  - Pan
  - Kitchen
  - Carte

THE SPOON UI

- Design and View Tabs
- Main Tree (Design): Lists all open transformations/jobs and their contents
- Run and Preview
- Execution Results
- Debugging

PDI CONFIGURATION FILES

User specific files found in .kettle folder in the User's home directory:

- kettle.properties: default properties file for variables
- shared.xml: default shared objects file
- repositories.xml: Local repositories file
- db.cache: database cache for metadata
- .spoonrc: User Interface setting, last opened ktrs/kjbs
- .languageChoice: User Language

INTRODUCTION TO TRANSFORMATIONS

- Manages Data Flow
- Smallest Unit
- Reads data from various sources, Transforms it and Loads it in various targets via steps
- Hops – Connector for various steps for a logical flow. Data and Metadata flow
- All the steps are initialized at once. Sequence not predictable.
- Processes the data as a stream. Can process unlimited no. of rows

CONNECTIONS

- Multiple connections can be created and shared
  - Connection can be shared by right-click and share
- Connection Types:
  - JDBC
  - ODBC
  - JNDI
- Driver jar is required in case of JDBC connection
- Database can be explored from the UI

INPUT STEPS

- These Steps pull the data from source
- Data is pulled in batches, size of which depends on the configuration
- Step Example:
  - Text File Input, CSV File Input, Fixed File Input
  - Table Input
  - Excel Input
  - Generate Rows
  - Get System Info

OUTPUT STEPS

- Used to Load the Data in the Target
- Commit Size can be configured (Table Output).
- Example Steps
    - Text Files Output
    - Table Output, Insert/Update, Delete, Update
    - Excel Output
    - XML Output

TRANSFORMATION STEPS

- Steps that are used to transform the data
- Example Steps:
  - Select Values
  - Calculator
  - Add Constants
  - Add Sequence, Add value fields changing sequence
  - Sort Rows
  - String Operations, String Cut
  - Unique Rows

LOOKUPS

- The Lookup feature in PDI accesses a data source to find values using defined matching criteria (a key value).
- Lookup Example:
  - Database Lookup
  - Stream lookup
  - Merge Join
  - Database join
  - Dimension Lookup/Update

SET TRANSFORMATIONS

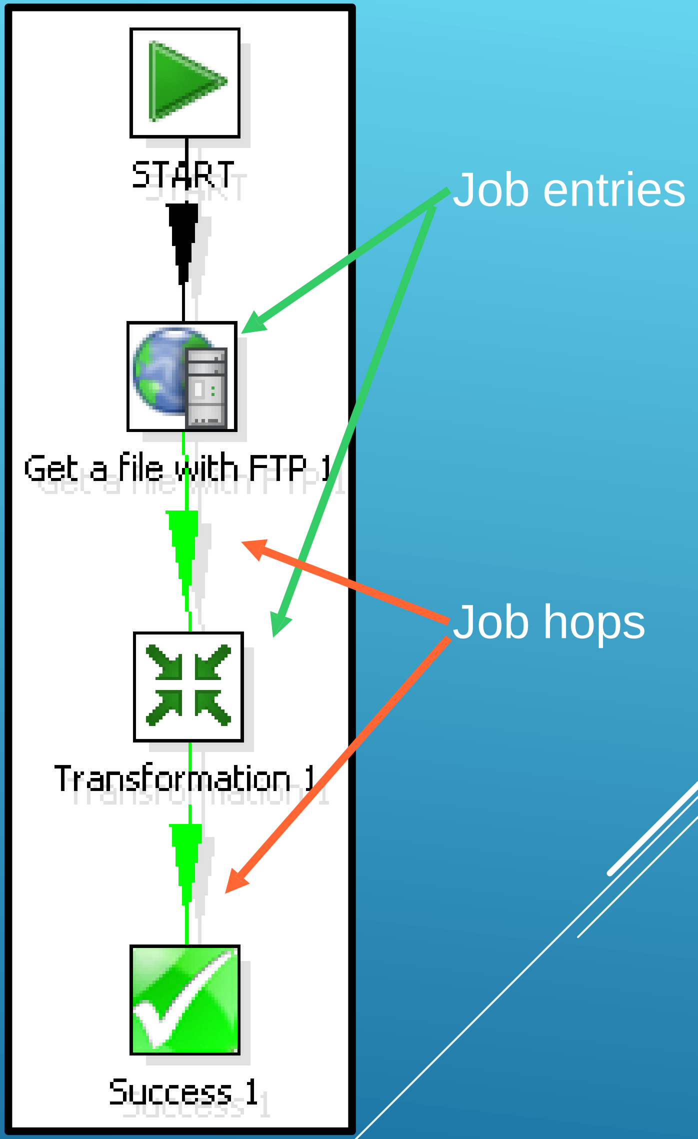
Set transformations are steps that operate on the entire set of data a stream.

The operations operate across all rows, not strictly within a row.

- Example Steps:
  - Filter rows
  - Sort Rows
  - Group by
  - Merge Rows
  - Unique Rows

INTRODUCTION TO JOBS

- Jobs aggregate individual pieces of functionality to implement an entire process.
- Manages the Work-flow of the ETL process
- Hops Types
  - Unconditional
  - Success – True Condition
  - Failure – False Condition
- Job Entries can be Job Entries, Transformations or another Job
- A Job starts with the START button



COMMONLY USED JOB ENTRIES

- Mail Section – Sends text or HTML mail with option of attachment.
- Scripting – Executes a script on the host where the job is running
  - Java Script
  - SQL Script
  - Shell
- Conditions – Checks or Evaluates some condition. The result is TRUE/FALSE.
  - Check DB Connection, Table Exists, Column Exists
  - Simple Evaluation
  - Check Folder is empty, File is locked, Check if files Exists, File Exists
- Utility – Some utilities for doing some tasks.
  - Abort Job
  - Ping a host, Telnet a host
  - Truncate table
  - Write to log

COMMONLY USED JOB ENTRIES CONTINUED…

- File Management – Basic File System task can be performed.
  - Create Folder, Delete Folder, Compare Folders
  - Create File, Delete File, Copy Files, Move Files
  - Unzip File, File Compare
- File Transfer – Move/Copies files from source to target via various protocols.
  - Get and Put File with FTP
  - Get and Put File with SFTP
  - Get and Put File with FTPS

ARGUMENTS, PARAMETERS AND VARIABLES

- **Arguments**
  - A PDI argument is a named, user-supplied, single-value input given as a command line argument (running a transformation or job manually from Pan or Kitchen, or as part of a script).
- Parameters
  - Parameters are like local variables; they are reusable inputs that apply only to the specific transformation that they are defined in. When defining a parameter, you can assign it a default value to use In the event that one is not fetched for it.
- Variables
  - A variable in PDI is a piece of user-supplied information that can be used dynamically and programmatically in a variety of different scopes.
  - PDI variables can be used in steps in both jobs and transformations

PENTAHO OLAP (MONDRIAN)

MONDRIAN OVERVIEW

- **Mondrian** is a ROLAP (relational online analytical processing) engine written in Java.
- Features of note:
  - MDX compliant
  - Supports aggregation, calculation and categorization of data from standard relational databases, including Oracle, MySQL, Microsoft SQL Server, IBM DB2, and others.
  - Automated Optimizations
    - Pre-compiled MDX expressions allow Mondrian server to optimize queries, choosing between evaluating an expression in memory and pushing the query to the RDBMS.
  - XML based cube schema
  - Aggregate Aware
  - Role Enabled

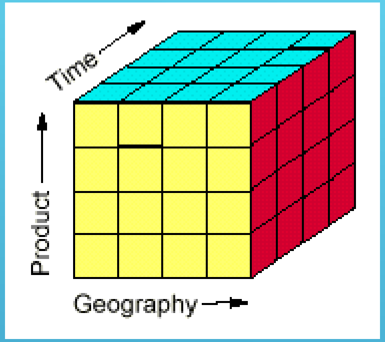
MONDRIAN AT WORK

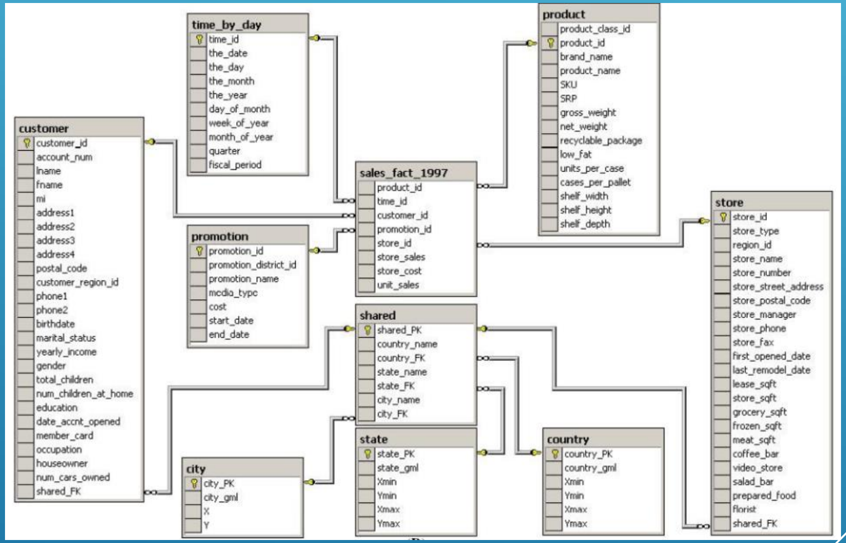
A dimensional model (logical)

- Cubes & virtual cubes
- Shared & private dimensions
- Calculated measures in cube and in query language
- Parent-child hierarchies

… mapped onto a star/snowflake schema (physical)

- Fact table
- Dimension tables
- Joined by foreign key relationships





HOW MONDRIAN SCHEMA MAPS TO DB SCHEMA

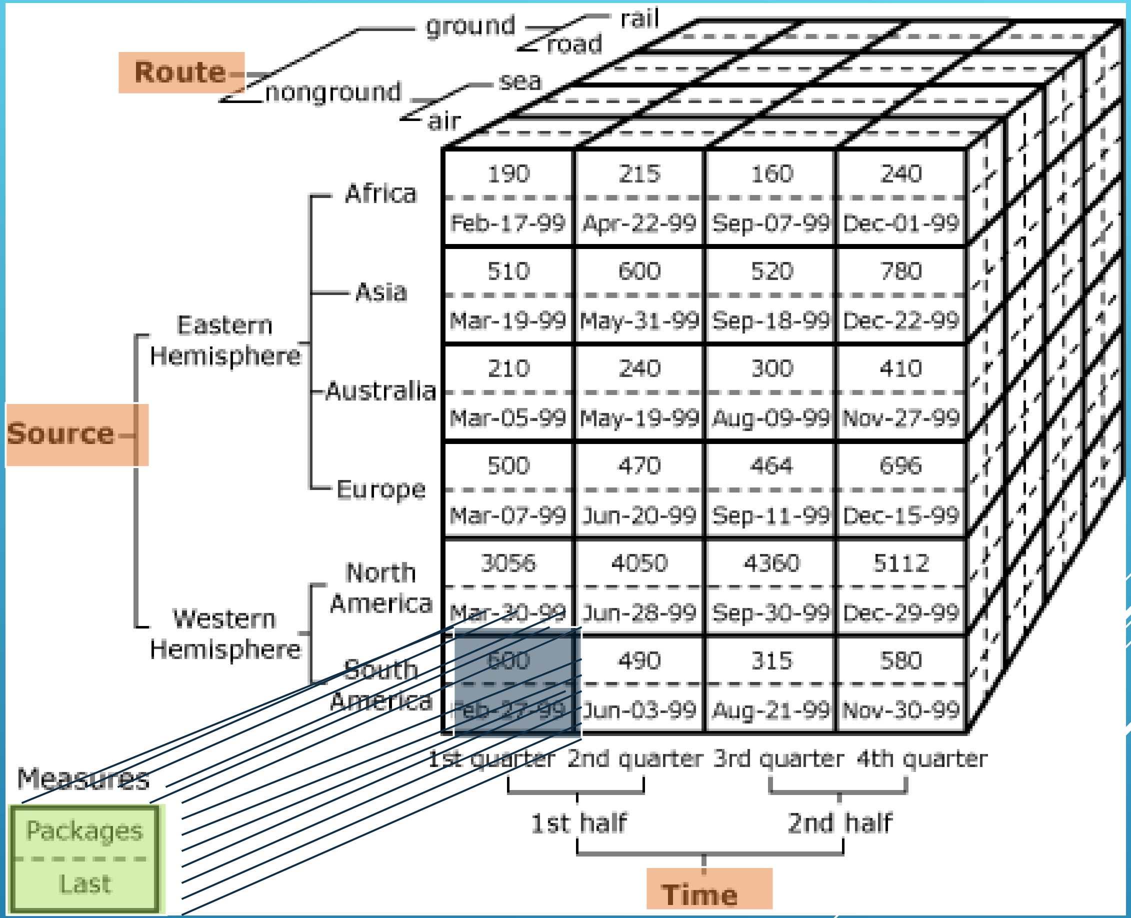
```
<Cube name=“moves">
    <Table name=“movementfact"/>
    <Dimension name="Terminal" foreignKey="terminal_id“>
        <Hierarchy name="Terminal" primaryKey="terminal_id">
          <Level name="Terminal" table="terminaldim“ column="terminal_name"/>
        </Hierarchy>
    </Dimension>
    <Measure name=“Miles" column=" miles“ aggregator="sum“ formatString="#,###"/>
    <Measure name=“Empty Miles" column=“empty_miles“ aggregator="sum“ formatString="#,###"/>
    <CalculatedMember name=“Empty Percent“ dimension="Measures“>
        <Formula>
            ([Measures].[Empty Miles] / [Measures].[ Miles]) * 100
        </Formula>
    </CalculatedMember>
 </Cube>
```

CUBES AND HIERARCHIES

- Relational databases organize data into flat, two-dimensional tables.
    - Rows & Columns with single data element intersections
- Multidimensional databases depend on structures called **cubes**.
- A cube is a collection of **measures** and **dimensions**
    - There can be n dimensions
    - Measures evaluated at the intersection of all n dimensions
    - Cubes can be sparse (minimal intersections) or dense (many intersections)
- Cubes also allow aggregation along dimensional **hierarchies**
    - Enables rapid drill-up/down
    - More flexible than table based construct

DIMENSIONS AND MEASURES

- Cubes can have more than two **dimensions**.
- The cube shown in the diagram has four *dimensions*: Route, Source, Time, and Measures.
- **Measures** represent the data organized by other dimensions included in the cube.
- The cube shown in the diagram has two *measures*, Packages and Last.



	1st quarter	2nd quarter	3rd quarter	4th quarter
Africa	190 Feb-17-99	215 Apr-22-99	160 Sep-07-99	240 Dec-01-99
Asia	510 Mar-19-99	600 May-31-99	520 Sep-18-99	780 Dec-22-99
Australia	210 Mar-05-99	240 May-19-99	300 Aug-09-99	410 Nov-27-99
Europe	500 Mar-07-99	470 Jun-20-99	464 Sep-11-99	696 Dec-15-99
North America	3056 Mar-30-99	4050 Jun-28-99	4360 Sep-30-99	5112 Dec-29-99
South America	600 Feb-27-99	490 Jun-03-99	315 Aug-21-99	580 Nov-30-99

🟫 → Dimension

🟩 → Measure

MONDRIAN CUBE CONSTRUCT

- A basic Mondrian Cube is structured in XML like the following

```
<Cube>
    <Dimension>
      <Hierarchy>
            <Level>
                  <Property/>
            </Level>
      </Hierarchy>
    </Dimension>
    <Measure/>
    <CalculatedMember/>
</Cube>
```

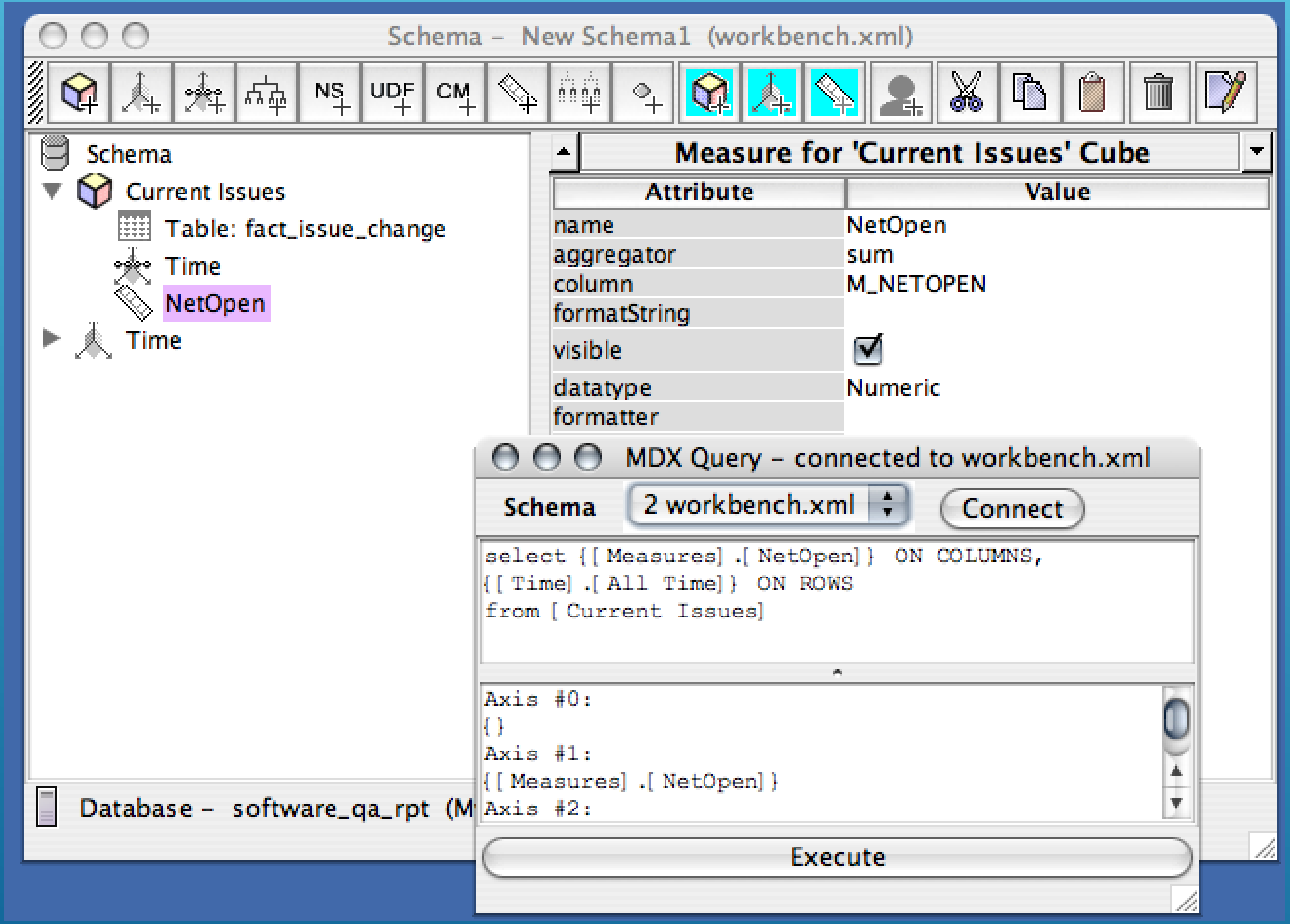
- It is possible to map to snowflake and similar relational structures.
- For complete schema reference: http://mondrian.pentaho.com/documentation/schema.php

FACT JOIN OPTIMIZATION

- The table mapping in the schema tells Mondrian how to get the data, but Mondrian is smart enough not to read the schema literally.
- It applies a number of optimizations when generating queries:
  - **If a dimension has a small number of members**, Mondrian reads it into a cache on first use.
  - **If a dimension is in the fact table** (or, more precisely, the level of the dimension being accessed is in the fact table), Mondrian does not perform a join.
  - **If two dimensions access the same table via the same join path**, Mondrian only joins them once.

SCHEMA WORKBENCH

- This is the tool to build and test the Mondrian schema



Schema – New Schema1 (workbench.xml)

Measure for 'Current Issues' Cube

Attribute	Value
name	NetOpen
aggregator	sum
column	M_NETOPEN
formatString	
visible	✓
datatype	Numeric
formatter	

Database – software_qa_rpt (M

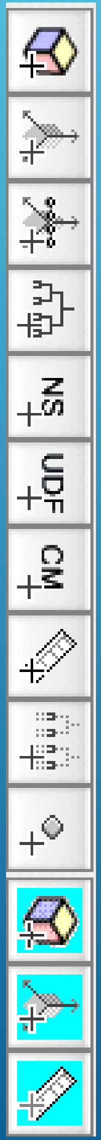
MDX Query – connected to workbench.xml

Schema: 2 workbench.xml

```
select {[Measures].[NetOpen]} ON COLUMNS,
{[Time].[All Time]} ON ROWS
from [Current Issues]
```

```
Axis #0:
{}
Axis #1:
{[Measures].[NetOpen]}
Axis #2:
```

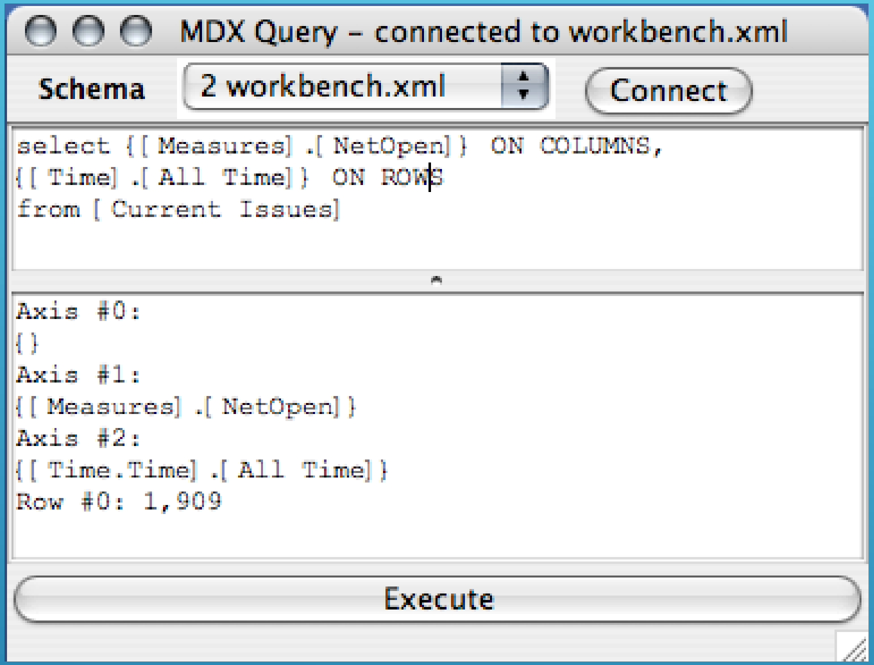
SW TOOLBAR



- Add Cube
- Add Dimension
- Add Dimension Usage
- Add Hierarchy
- Add Named Set
- Add User Defined Function
- Add Calculated Member
- Add Measure
- Add Level
- Add Property
- Add Virtual Cube
- Add Virtual Dimension
- Add Virtual Measure

MDX IN PSW

- Must have a JDBC connection setup
- Select a "schema"
  - List is the Schemas windows that are currently open in the application
  - Connect to verify the Schema/connection
- Top Window
  - Enter in an MDX query
- Bottom Window
  - Review your results after hitting Execute
  - NOTE: Format is a text printout, NOT a crosstab



```
select {[Measures].[NetOpen]} ON COLUMNS,
{[Time].[All Time]} ON ROWS
from [Current Issues]
```

```
Axis #0:
{}
Axis #1:
{[Measures].[NetOpen]}
Axis #2:
{[Time.Time].[All Time]}
Row #0: 1,909
```

MDX AT WORK

- **MDX** stands for 'multi-dimensional expressions'.
- It is the main query language implemented by Mondrian.
- This was developed by Microsoft.
- MDX is used to
  - Query a multi-dimensional data source to return values.
  - Navigate the structure of an OLAP data source (cube).
  - Obtain information about dimension hierarchy and its members.
  - Create a calculated member via MDX expressions.

MDX SYNTAX

- MDX and SQL are similar in several aspects, including the fact that they both use a few common key words to perform queries such as
  - **SELECT** clause (Tuple) ON COLUMNS/ON ROWS,
  - **FROM** clause (which cube), and
  - **WHERE** clause (slicer).
- A basic MDX query looks like this:

```
SELECT {([Measures].[sales], [Measures].[profit])} ON COLUMNS, {([Region],
 [product])} ON ROWS
 FROM [salescube]
 WHERE ( [Time].[2016] )
```

- Mondrian specific MDX documentation is here:

http://mondrian.pentaho.com/documentation/mdx.php

CELLS, TUPLES AND SETS

- Tuples uniquely identify individual cells or groups of cells (a cube slice)
    - Tuples are enclosed with ()
    - Example Cube Slice: profit for North Region in the Region in Q1 of 2016
        - ([Region].[North Region], Time.[2016].[1], profit)
- Sets are ordered collections of tuples
    - Sets are enclosed by {}
    - All Tuples in a set must have the same dimensionality
    - A set can contain a single tuple
    - Example Set: Profit for 2016 and 2015
        - {(Time.[2016], Profit) , (Time.[2015], Profit)}
- For those interested in more detailed MDX reference, check out this site:
    - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/olapdmad/agmdxbasics_04qg.asp

SECURITY

- In Pentaho, security can be implemented in two ways with Mondrian
  - Role based Security (Static)
  - Using DSP (Dynamic Schema Processor)
- We will cover both the approaches in the training

ROLE BASED SECURITY

- A **role** defines an access-control profile
- It has a series of grants (or denials) for schema elements
- Roles are set when establishing a Mondrian connection
- Roles are defined by **<Role>** elements, which occur as direct children of the **<Schema>** element, after the last **<Cube>**

```
<Role name=“South Manager">
  <SchemaGrant access="none">
    <CubeGrant cube=“sales" access="all">
      <HierarchyGrant hierarchy="[Region]" access="custom"
topLevel="[Region].[Territory]“ bottomLevel="[Region].[Territory].[State].[City]“ >
        <MemberGrant member="[Region].[South Region]" access="all"/>
      </HierarchyGrant>
     </CubeGrant>
  </SchemaGrant>
</Role>
```

- The above example restricts South Manager to seeing only South in the Sales cube

SECURITY CONTINUED…

- Mondrian Roles are mapped with the BA server in the following ways:
  - One-To-One UserRoleMapper
  - LookupMap UserRoleMapper
  - UserSession UserRoleMapper
- Sample:
- <**Role** name="California manager">
  - <SchemaGrant access="none">
    - <CubeGrant cube="Sales" access="all">
      - <HierarchyGrant hierarchy="[Store]" access="custom" topLevel="[Store].[Store Country]">
        - <MemberGrant member="[Store].[USA].[CA]" access="all"/>
        - <MemberGrant member="[Store].[USA].[CA].[Los Angeles]" access="none"/>
      - </HierarchyGrant>
      - <HierarchyGrant hierarchy="[Customers]" access="custom“ topLevel="[Customers].[State Province]" bottomLevel="[Customers].[City]">
        - <MemberGrant member="[Customers].[USA].[CA]" access="all"/>
        - <MemberGrant member="[Customers].[USA].[CA].[Los Angeles]"access="none"/>
      - </HierarchyGrant>
      - <HierarchyGrant hierarchy="[Gender]" access="none"/>
    - </CubeGrant>
  - </SchemaGrant>
- </Role>

DYNAMIC SCHEMA PROCESSOR

- A Dynamic Schema Processor (DSP) is a special class that can intercept the schema when it is loaded.
- It has access to the session variables and hence can modify the schema based such as user ID, roles and tenant ID
- More details on the implementation of the same can be found at the following locations:
  - http://mondrian.pentaho.com/documentation/schema.php#Schema_processor
  - http://help.pentaho.com/Documentation/6.1/0R0/070/Multi-Tenancy . Here go to **Analyzer Data Multi-Tenancy** section for more details.

PRIVILEGE HIERARCHY

- Privileges are established and overridden along the following hierarchy:
- Schema
  - Cube
    - Hierarchy
      - Member
- Establishing access defaults at higher levels and overriding those defaults among the lower levels

SCHEMA AND PRIVILEGE

- A **<SchemaGrant>** defines the default access for objects in a Schema.
  - The access attribute can be "all" or "none“.
  - This default access can be overridden by Cube in the Schema.
- A **<CubeGrant>** defines the default access to a particular cube within the Schema.
  - The access attribute can be "all" or "none“.
  - The access can be overridden by Hierarchies in the Cube.

HIERARCHY PRIVILEGE

- A **<HierarchyGrant>** defines access to Hierarchy within a Cube.
- Access can be set to “all”, “none” or “custom”
- “all” and “none” follow the pattern as before
- "custom”, enables additional attribute usage for the Hierarchy
  - “topLevel” can be set to identify the highest level of member aggregation that the user can access (preventing view of “big picture”)
  - “bottomLevel” can be set to identify the lowest level of member detail that the user can access (preventing view of “private details”)
  - **<MemberGrant>** elements may be nested to identify member specific access within the Hierarchy

MEMBER PRIVILEGE

- **<MemberGrant>** gives (or removes) access to a given member, and all of its children. Here are the rules:
  - **Members inherit access from their parents**.
    - If you deny access to parent, you won't be able to see its children.
  - **Grants are order-dependent**.
    - If you grant access to a parent, then deny access to one of the children, then you will be able to see all other children except the one that was denied.
    - If you deny access to one of the children, then grant access to the parent, you would override and have access to the child.
  - **A member is visible if any of its children are visible**.
    - Suppose you deny access to a parent, then grant access to one of its children, then you would override and gain access to the parent summary but none of the detailed siblings of the child.
  - **Member grants don't override the hierarchy grant's top- and bottom-levels**.

CONNECTIONS AND ROLES

- A role only has effect when it is associated with a Mondrian connection.
- By default, connections have a role which grants them access to every cube in that connection's schema.
- Most databases associate roles (or 'groups') with users, and automatically assign them when users log in.
- However, Mondrian does not have the notion of users, so you have to establish the role in a different way.
- Pentaho action sequences and components enable passing of the Role into the Mondrian server.

PERFORMANCE TUNING FOR ANALYSIS

- At the table level the DBA will help identify the tables to be optimized
- Mondrian can use Aggregate tables for better performance.
- Detail on how to implement and use aggregate tables can be found at

http://mondrian.pentaho.com/documentation/aggregate_tables.php

- There are certain rules that need to be followed to use aggregate tables.

CACHE

- By default, Mondrian maintains the cache for a query executed.
- This is based on connection and schema combination.
- This helps the roundtrips to the Database.
- But, will have the stale data if the underlying data has changed.
- Cache should be flushed on a regular basis by the Administrator.
- This can be done from the Pentaho BA server UI or by calling/scheduling an xaction.
- The cache can also be cleared programmatically by calling few APIs exposed by Mondrian.

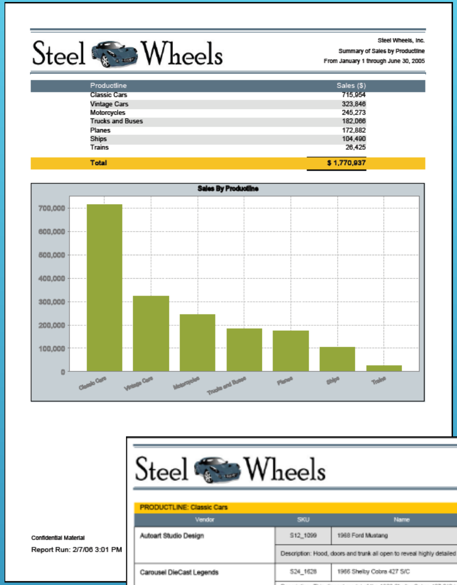
http://mondrian.pentaho.com/documentation/cache_control.php#CacheControl_API

PENTAHO REPORTING

- There are two tools for reporting in Pentaho
  - Report Designer
  - Interactive Reporting (based on Metadata Editor created models)

PENTAHO REPORT DESIGNER

- Started as an open source - Jfree Report project.
- Provides critical functionality for end users
  - Web-based Access
  - Prompting/Parameterized Reports
  - Scheduling
  - Subscriptions
  - Bursting/Distribution
  - Graphical Design Tools
- Provides features for developers
  - Heterogeneous Data Sources
    - Relational, OLAP (Mondrian), XML, custom and Pentaho Data Integration Transformations
  - Modular Report Definition
    - Separates presentation from query
  - Integration points to applications, portals
    - JSP, Portlet, Web Service

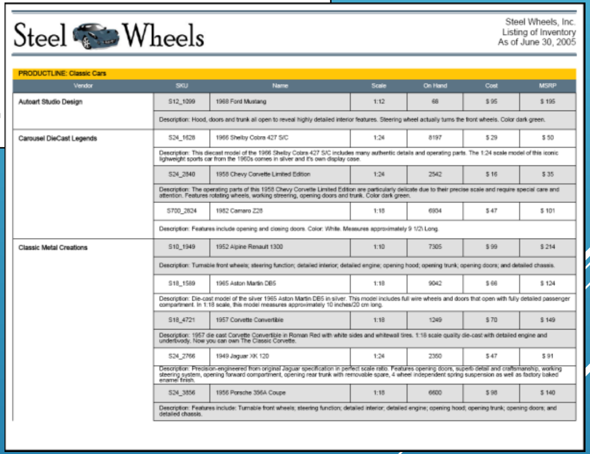


Steel Wheels

Steel Wheels, Inc.
Summary of Sales by Productline
From January 1 through June 30, 2005

Productline	Sales ($)
Classic Cars	715,954
Vintage Cars	323,846
Motorcycles	245,273
Trucks and Buses	182,066
Planes	172,882
Ships	104,490
Trains	26,425
Total	$ 1,770,937

Confidential Material
Report Run: 2/7/06 3:01 PM



Steel Wheels

Steel Wheels, Inc.
Listing of Inventory
As of June 30, 2005

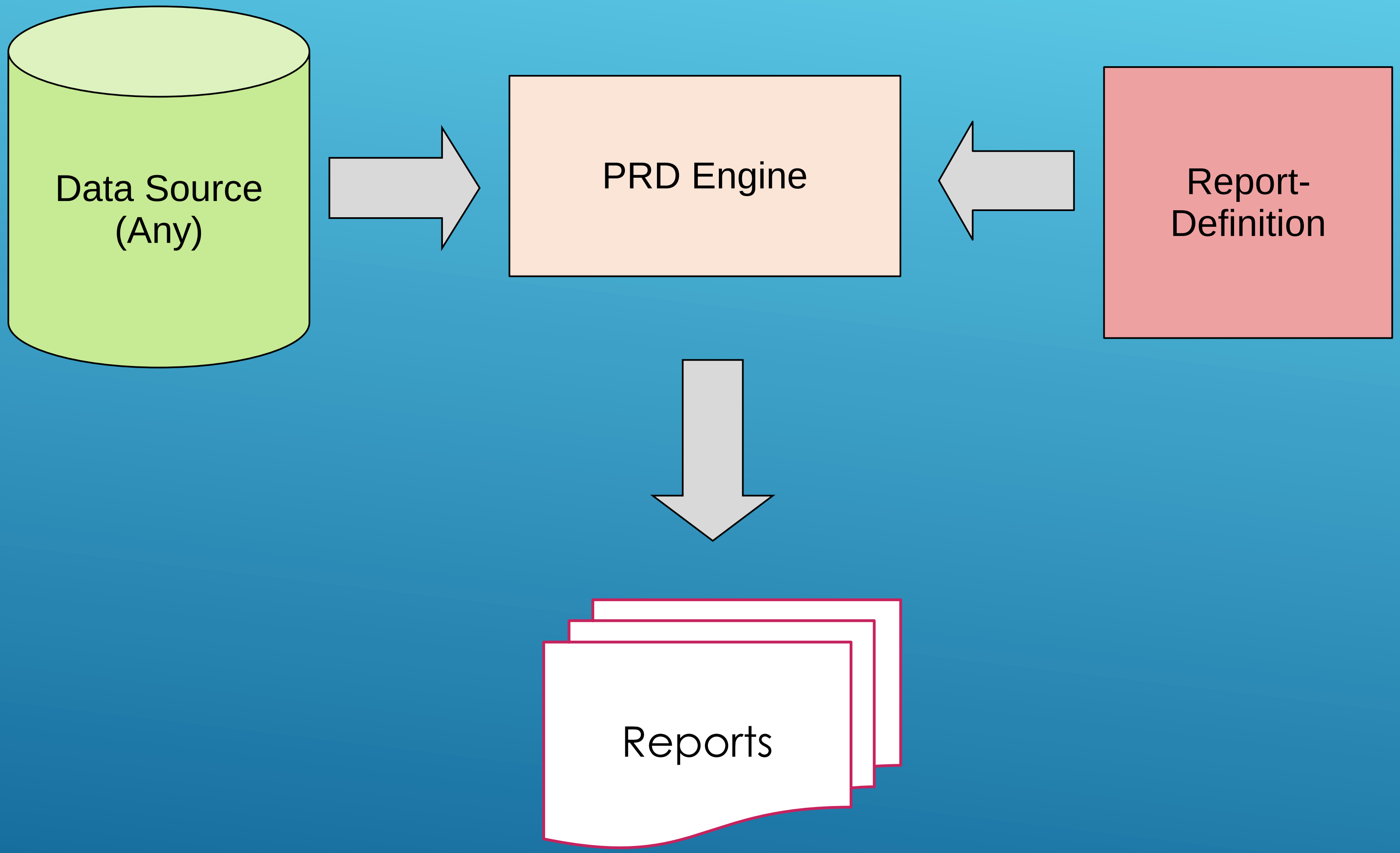
PRODUCTLINE: Classic Cars

Vendor	SKU	Name	Scale	On Hand	Cost	MSRP
Autoart Studio Design	S12_1099	1968 Ford Mustang	1:12	68	$ 95	$ 195
	Description: Hood, doors and trunk all open to reveal highly detailed interior features. Steering wheel actually turns the front wheels. Color dark green.					
Carousel DieCast Legends	S24_1628	1966 Shelby Cobra 427 S/C	1:24	8197	$ 29	$ 50
	Description: This diecast model of the 1966 Shelby Cobra 427 S/C includes many authentic details and operating parts. The 1:24 scale model of this iconic lighweight sports car from the 1960s comes in silver and it's own display case.					
	S24_2840	1958 Chevy Corvette Limited Edition	1:24	2542	$ 16	$ 35
	Description: The operating parts of this 1958 Chevy Corvette Limited Edition are particularly delicate due to their precise scale and require special care and attention. Features rotating wheels, working streering, opening doors and trunk. Color dark green.					
	S700_2824	1982 Camaro Z28	1:18	6934	$ 47	$ 101
	Description: Features include opening and closing doors. Color: White. Measures approximately 9 1/2\ Long.					
Classic Metal Creations	S10_1949	1952 Alpine Renault 1300	1:10	7305	$ 99	$ 214
	Description: Turnable front wheels; steering function; detailed interior; detailed engine; opening hood; opening trunk; opening doors; and detailed chassis.					
	S18_1589	1965 Aston Martin DB5	1:18	9042	$ 66	$ 124
	Description: Die-cast model of the silver 1965 Aston Martin DB5 in silver. This model includes full wire wheels and doors that open with fully detailed passenger compartment. In 1:18 scale, this model measures approximately 10 inches/20 cm long.					
	S18_4721	1957 Corvette Convertible	1:18	1249	$ 70	$ 149
	Description: 1957 die cast Corvette Convertible in Roman Red with white sides and whitewall tires. 1:18 scale quality die-cast with detailed engine and underbvody. Now you can own The Classic Corvette.					
	S24_2766	1949 Jaguar XK 120	1:24	2350	$ 47	$ 91
	Description: Precision-engineered from original Jaguar specification in perfect scale ratio. Features opening doors, superb detail and craftsmanship, working steering system, opening forward compartment, opening rear trunk with removable spare, 4 wheel independent spring suspension as well as factory baked enamel finish.					
	S24_3856	1956 Porsche 356A Coupe	1:18	6600	$ 98	$ 140
	Description: Features include: Turnable front wheels; steering function; detailed interior; detailed engine; opening hood; opening trunk; opening doors; and detailed chassis.					

WHY PRD?

- Flexible Report Design
  - Graphical design environment
  - Templates for consistent report formats
  - Merges for commonly used report elements
  - Access relational, OLAP, or XML data
  - Conditional hiding of report objects
- Embeddable
  - Lightweight
  - Easy to extend
- 100% Java - Portability, scalability, integration
- Cross Platform (client and server) - Mac, Linux/Unix, Windows
- Multiple Output formats – HTML, CSV, PDF, Excel and Word

ARCHITECTURE



REPORT DISTRIBUTION

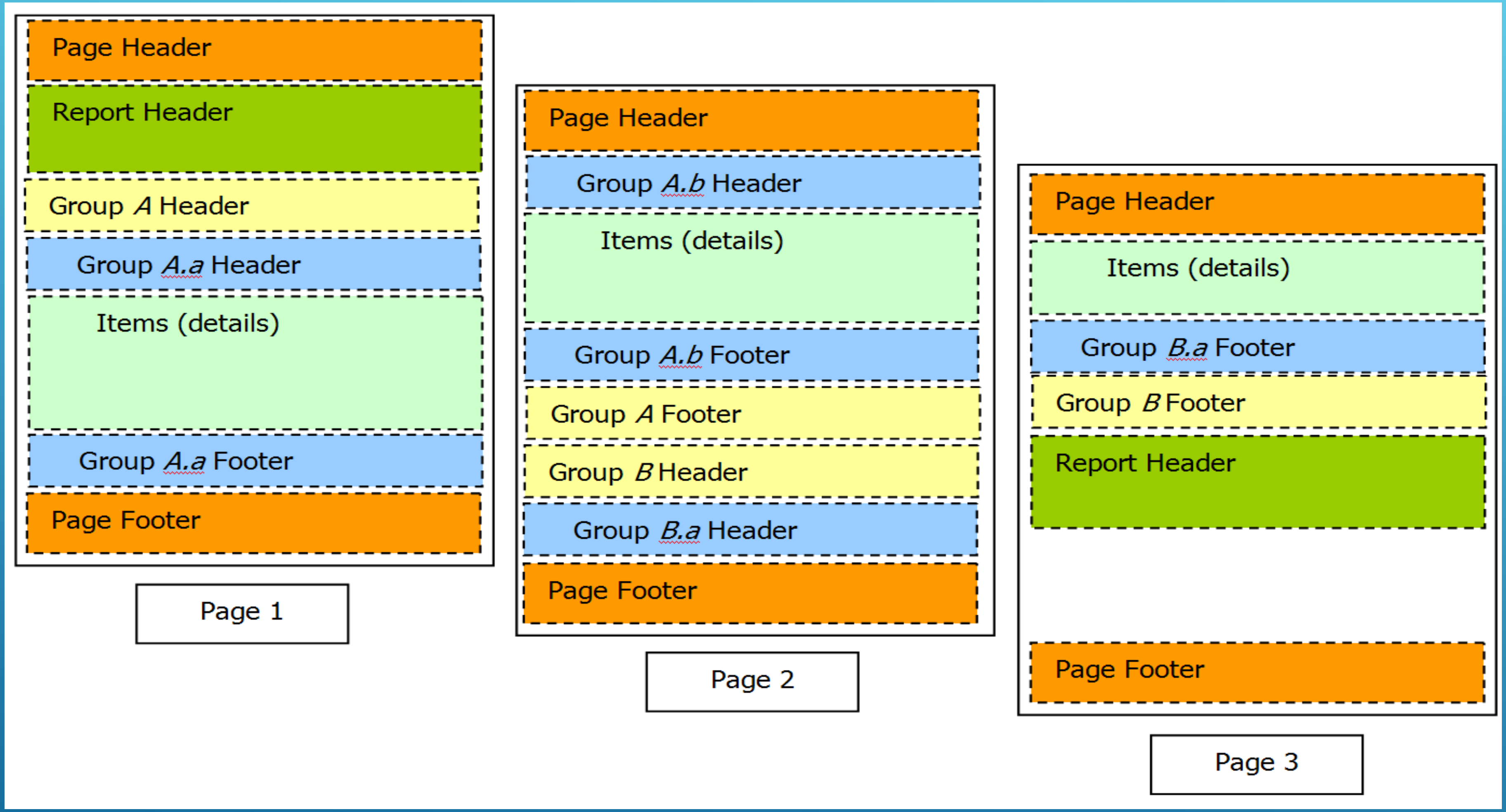
Publishing Report to Reporting Server's Solution repository allows it to be:

- Viewed online via intranet or internet
- Viewed online as a portlet within a portal
- Burst
- Scheduled/Subscribed

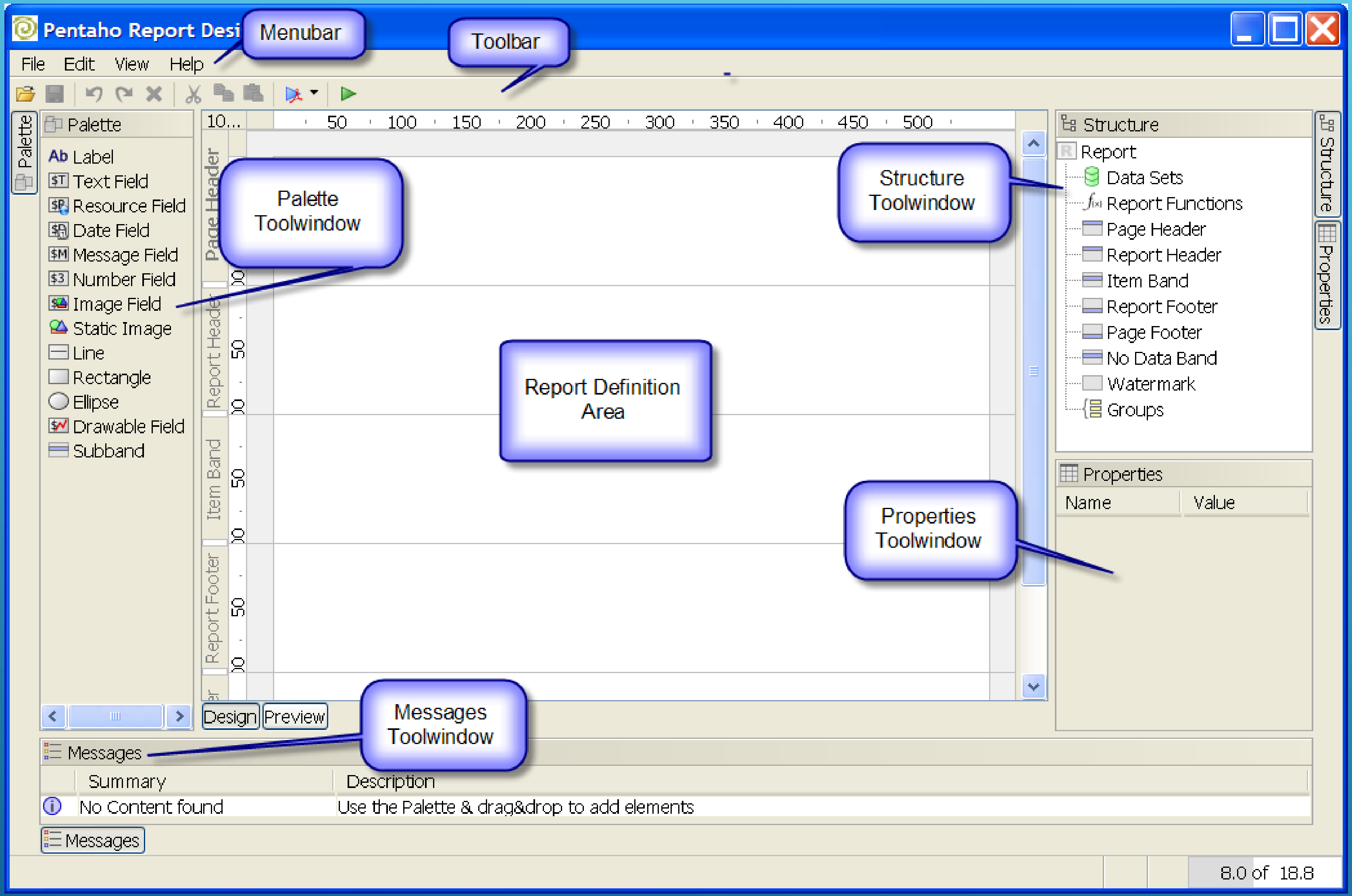
REPORT OBJECTS

- Data Fields (textual, numeric, date)
- Message Field (mix all data fields type and other textual content)
- Labels (text field)
- Shapes
- Rectangles (used for row or column banding)
- Bands (group of objects)
- Lines
- Images
- Charts
- Calculated Fields (Functions/Expressions)
- Custom Calculated Fields (BeanShell Expression)

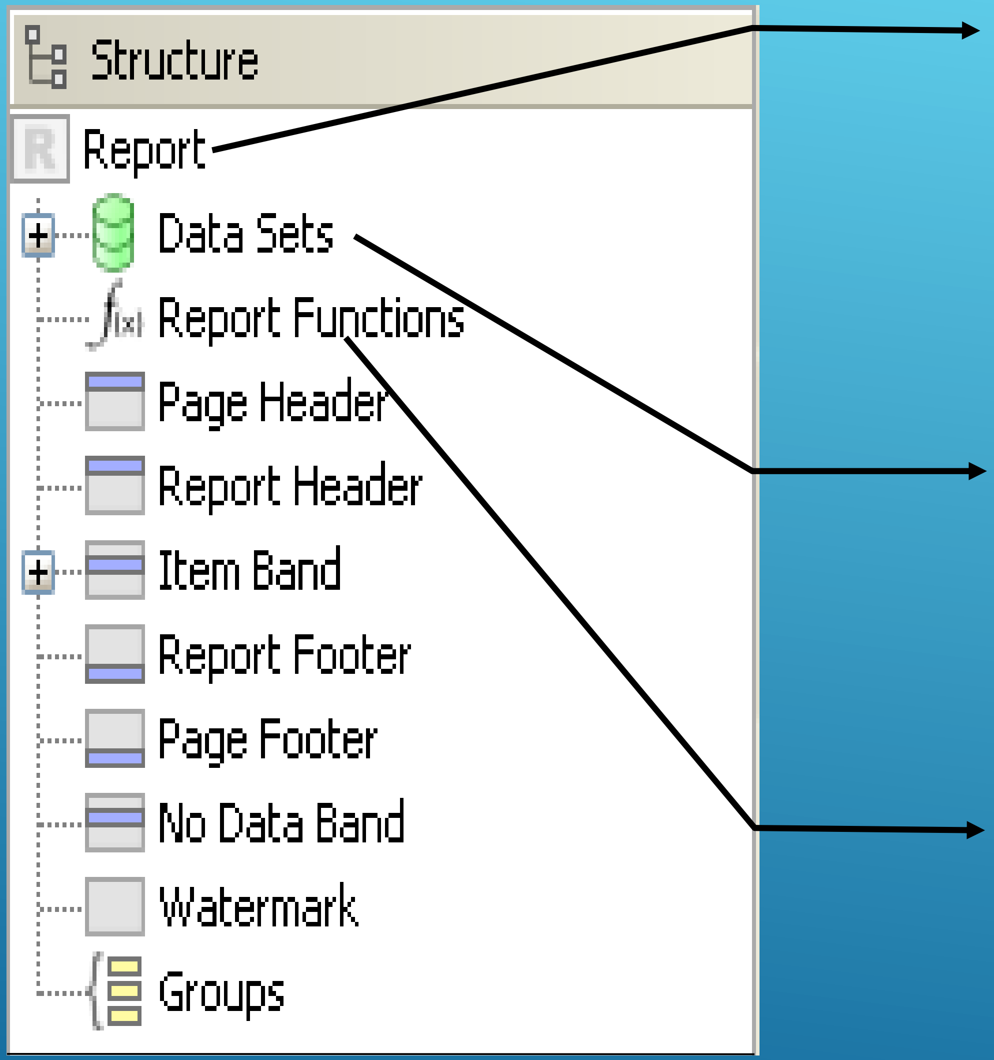
REPORT SECTIONS



PRD LAYOUT



Summary	Description
No Content found	Use the Palette & drag&drop to add elements



- **Report**

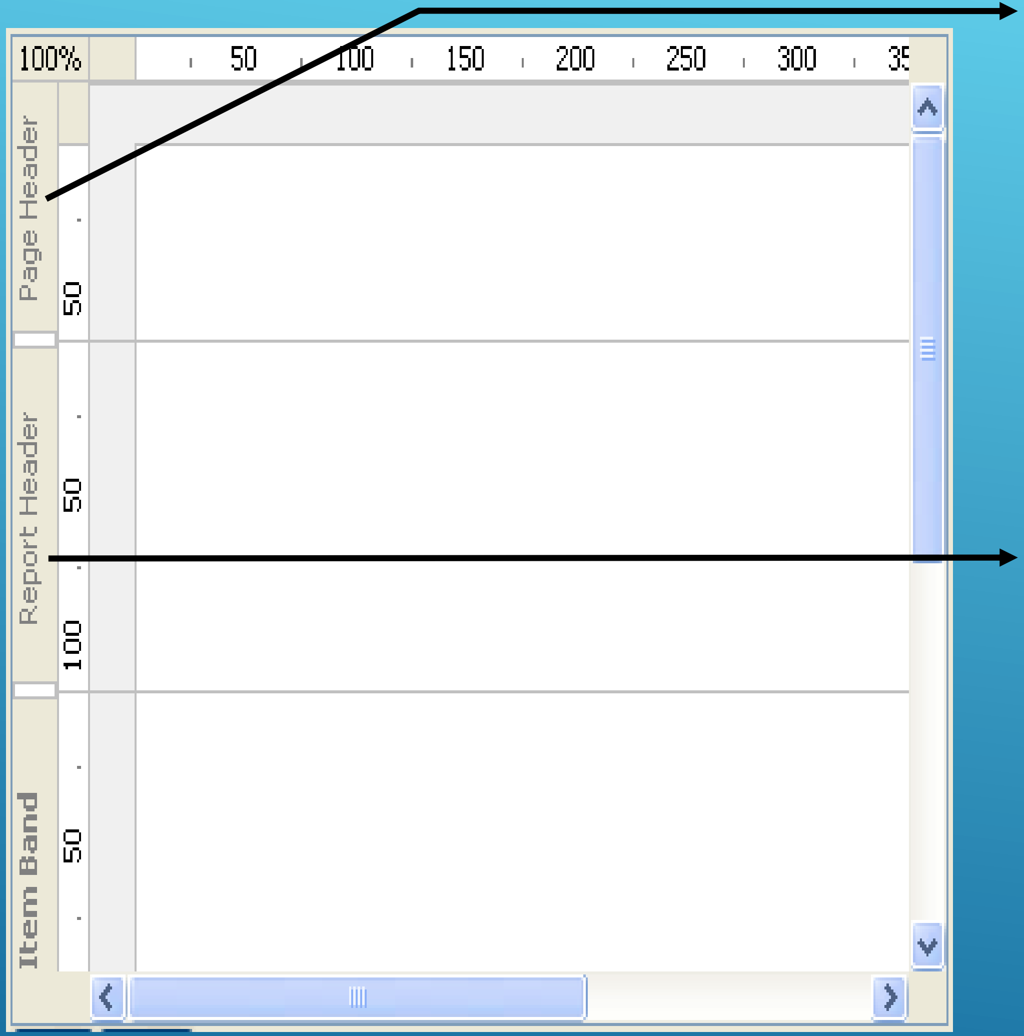
  This is the master node that will have all the elements present in the report.

- **Data Sets**

  Any dataset that is added to the report will be present here.

- **Report Functions**

  Functions that can be used on the elements are listed under this.

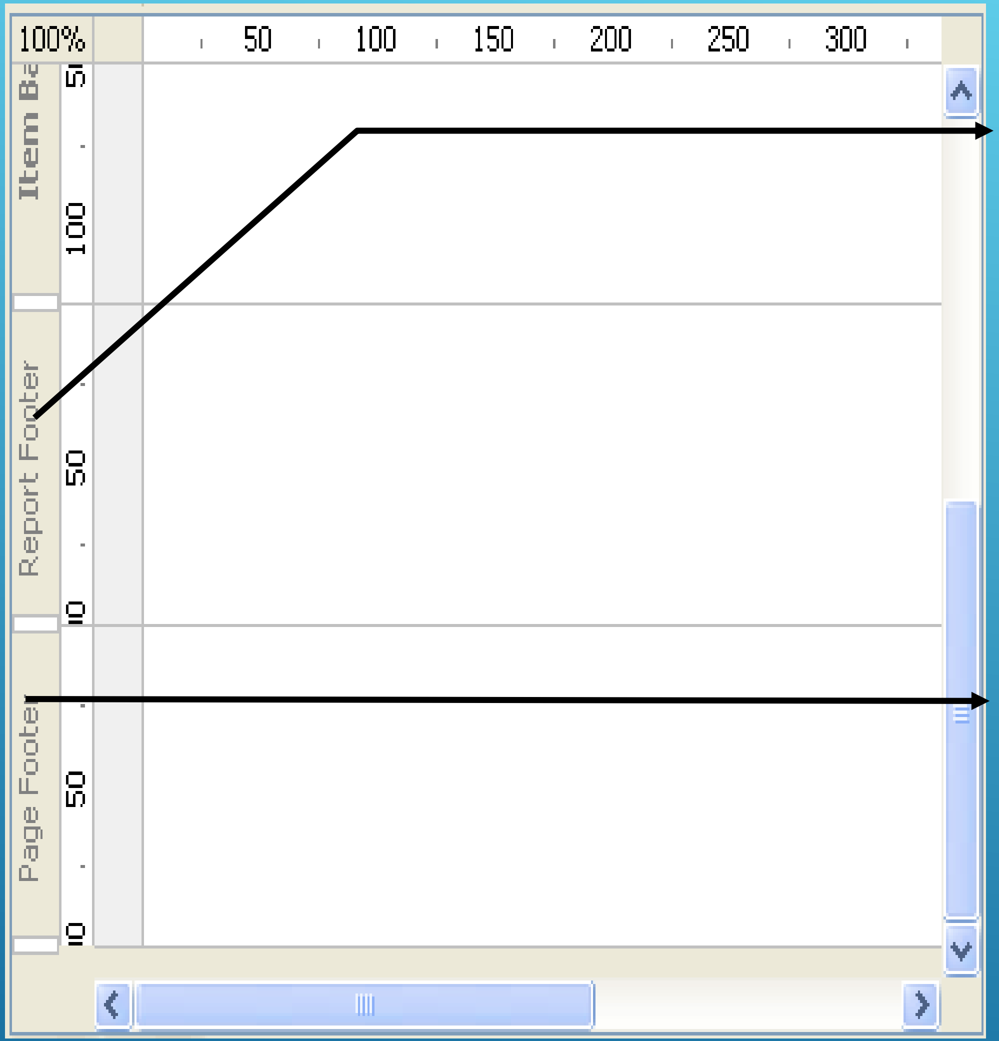


- **Page Header Band**

  The content in this band will be present in each of the output pages of the report. This can be configured to print only on the 1st and the last page.

- **Report Header Band**

  After the page header, this will be printed only once; at the beginning of the report.

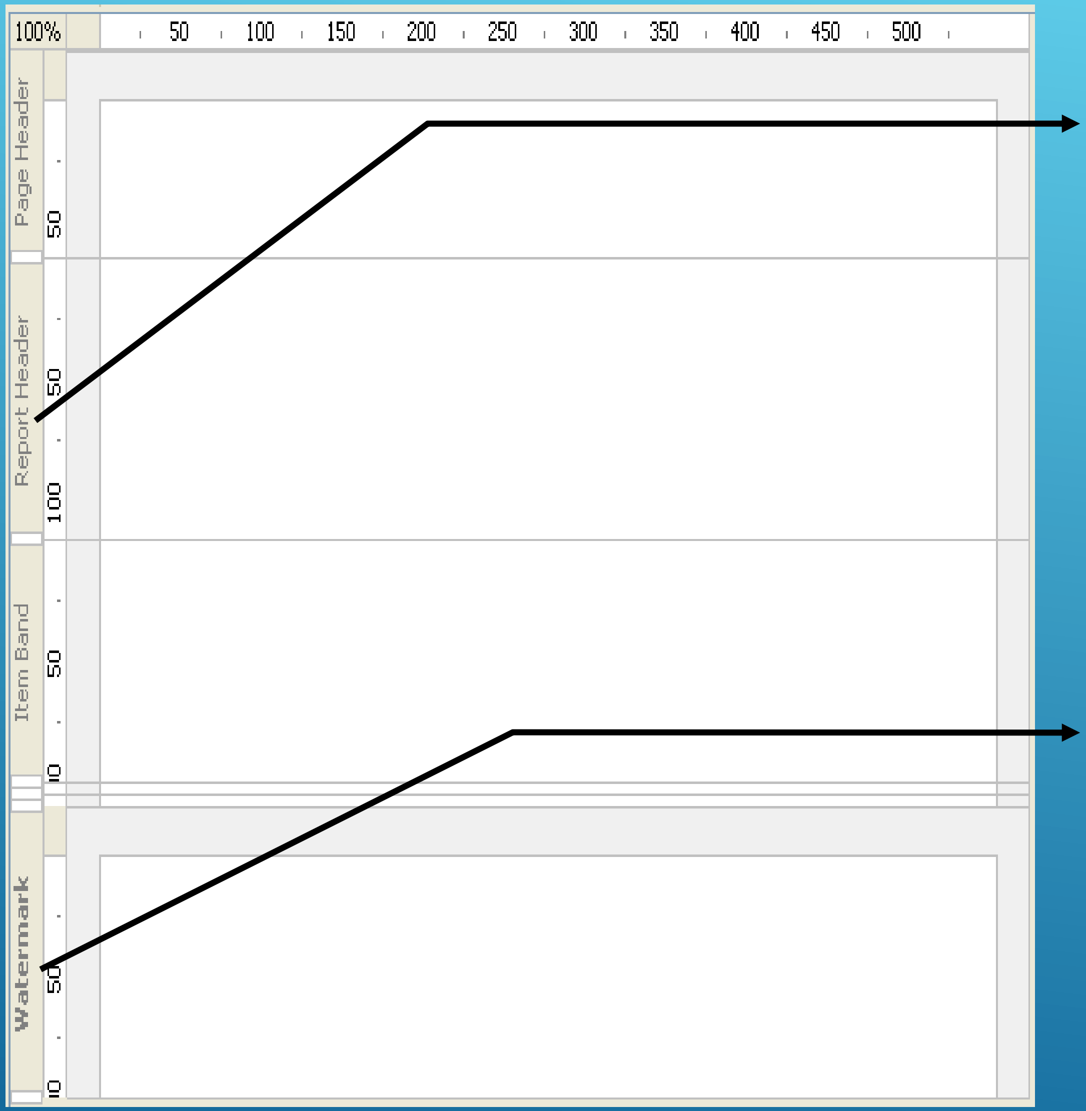


- **Report Footer Band**

  If not empty, the content of this section will be printed at the end of the report just before the page footer.

- **Page Footer Band**

  The page footer band, when not empty, is printed at the bottom of each page. This can be configured to print only on the 1st and the last page.

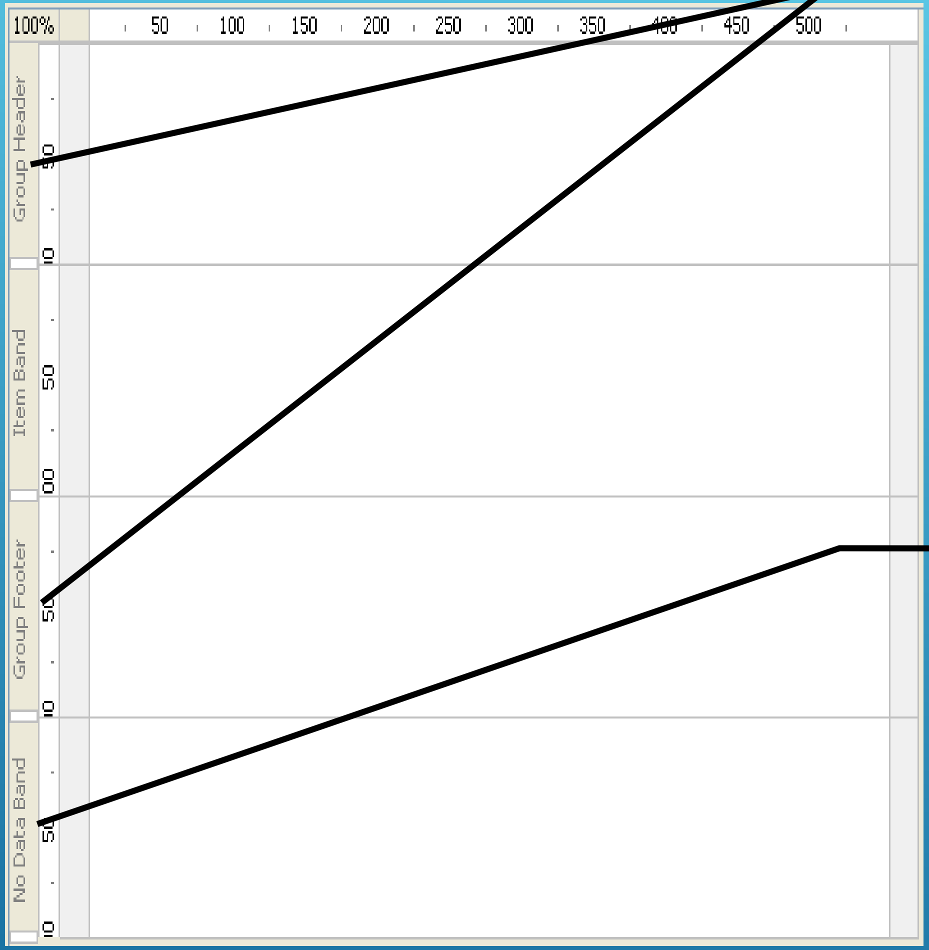


- **Item Band**

  This section will have the data to be displayed by the Data set queries. Only one Query can be called in this section at a time.

- **Watermark Band**

  The watermark band is printed on the background of every page.



- **Groups**

  This band will have the group and sub-group's details. All the groups will appear here.

- **NoData Band**

  The content of this band will be displayed when no Data is returned from the Data –set queries.

CONNECTIONS

- There are various ways to connect to a Data Source in PRD:
  - JDBC – Needs a driver corresponding the Database
  - JNDI – Configured at the BA server.
  - ODBC – ODBC connections
  - PDI transformations – a KTR can be a Data Source which in turn can connect to a lot more Data Sources.
  - Pentaho OLAP Schema – The OLAP schema along with a database connection can act as a source for the Report.
  - Pentaho Metadata Model – The Metadata model can be used as a source for the report.

REPORT FUNCTION

- A **function** is a custom program that can return a value depending on other values available in the report.
- It can use values available in a dataset or use the value returned by another function.
  - *Functions are often parameterized by outputs of other functions*
- Report Functions can be used to:
  - Format report content (row banding, remove dups, conditional display, etc)
  - Add hyperlinks
  - Generate row level calculations
  - Calculate group and report level aggregations (e.g. sum, average, min, max)
  - Add and format charts
- Functions can be added using the **Data tab** by selecting the **Functions** node. All available functions are listed in the **Add Function** window.
- It can also be added to a particular element by using the + on the right side of the element.

HYPERLINK

Function Class	Description
CreateHyperLinksFunction	Adds hyperlinks to all elements with the name specified in 'element'. The link target is read from a specified field. The column referenced by this field should contain URLs or Strings.
TextFormatExpression	Uses a java.text.MessageFormat to concatenate and format one or more values evaluated from an expression, function or report data source.

HOW TO ADD HYPERLINK

- Identify the field that will become a hyperlink and specify its element name with something meaningful (e.g. URLFIELD)
- Create a TextFormatExpression report function to define the url giving it a meaningful name (e.g. URLTEXT). If the url must be dynamically determined based on report fields, use those and refer to them using the {<n>} variable replacement syntax. <n> refers to the number of the variable field specified starting with 0 for the first field.
- Create a CreateHyperLinksFunction report function whose element is named the same as your field element name (e.g. URLFIELD) and whose field is named the same as your TextFormatExpression (e.g. URLTEXT).

GROUPING

- Groups can be added in the **Structure tool-window** by selecting the **Groups** node, opening the popup menu and selecting **Add Group.**
- To delete a group, select **Groups** node in **Structure tool-window** and press the **Delete** key.
- When you expand the **group** in the **Structure tool-window** you can see that there is also a **group header band** and a **group footer band** available.
- Each time a new group starts, the **group header band** of this group is printed. Whenever a group ends, the **group footer band** is printed.
- Subgroups are nested within their parents. You must expand the parent to see the children.
- Group header and footer bands do not appear in the graphical report definition area unless you make them appear by selecting the bands in the **Structure tool-window** and turning the **Show In Layout GUI** property on.

REPORT PARAMETERIZATION

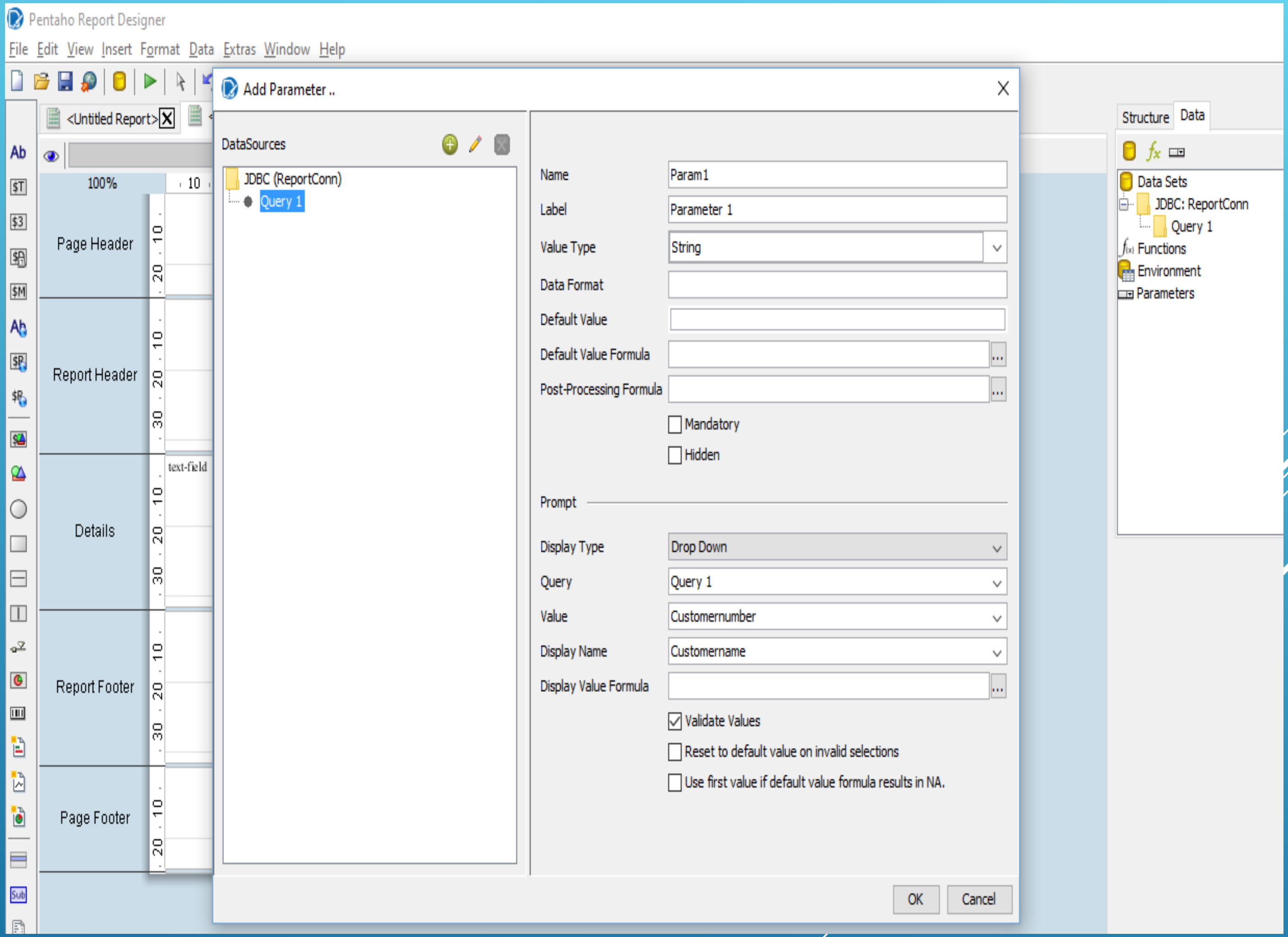
- Reports may be parameterized in one of two general ways
  - Query
  - Report "behaviour"
- Query Parameters…
  - Change what data is provided to the report for execution
- Report Behaviour Parameters can…
  - Change how report functions operate, thus
    - changing what elements are displayed and/or
    - changing the display format of elements and/or
    - changing the formulas by which fields are calculated and/or
    - etc…

HOW TO DO IT?

- Configure the Parameter:
  - Add a Parameter from the Data Tab in the report.
  - Add/Use the Datasource.
  - Select the appropriate options for the parameter.
- The above defined parameter can be passed on to the main query to filter the dataset.
- Use the parameter in the query like the following example:

  Select productname, city, amount from sales

  where customernumber = ${Param1}



Add Parameter ..

DataSources
- JDBC (ReportConn)
  - Query 1

Name	Param1
Label	Parameter 1
Value Type	String
Data Format	
Default Value	
Default Value Formula	
Post-Processing Formula	
	☐ Mandatory
	☐ Hidden

Prompt

Display Type	Drop Down
Query	Query 1
Value	Customernumber
Display Name	Customername
Display Value Formula	
	☑ Validate Values
	☐ Reset to default value on invalid selections
	☐ Use first value if default value formula results in NA.

OK Cancel

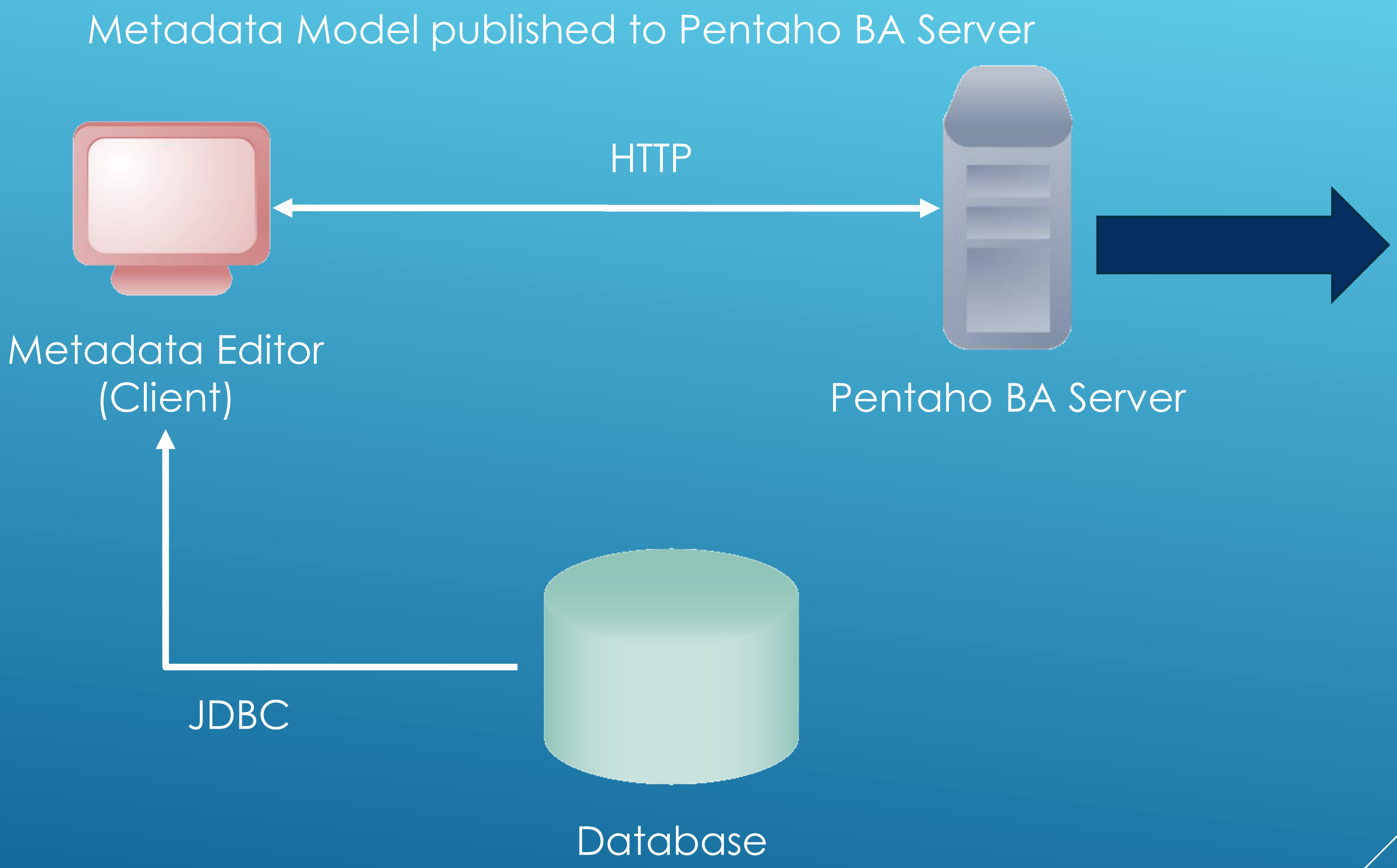
SUB REPORTS

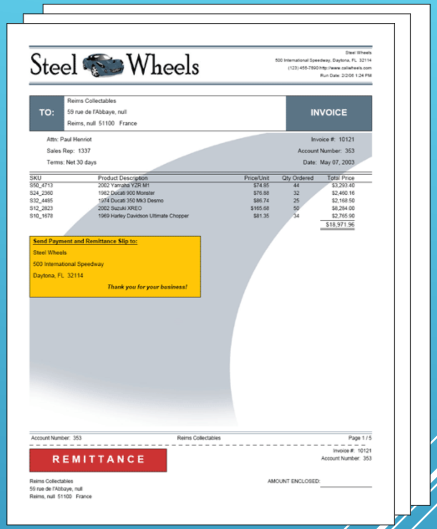
- A sub report is a fully functional report that can be added to other reports the same way other report elements are added.
- Sub report can be placed in any band of the “Master” report.
- Sub reports generally use a different query than the “Master” report therefore you must create 2 data sources within the “Master” report.
- It can make use of the inherited Data sources or can have one itself.
- It can also be parameterized as the main report. The only difference is, it needs to import the parameter from the “Master” report.

INTERACTIVE REPORTING

- Removes the need to know query languages
  - Generates query based upon metadata selection
  - Query generated at runtime
- Centralized maintenance
  - Can be edited by client tools
  - Available to entire solution
  - Changes reflected at runtime
- Integration with existing metadata systems
- Security enforced at content design time and runtime
- Embeddable and Extendible
  - Metadata can be queried programmatically
  - Also exists as an embeddable Java component
  - Open and extendible architecture

PENTAHO METADATA ARCHITECTURE





Steel Wheels

Steel Wheels
500 International Speedway, Daytona, FL 32114
(123) 456-7890 http://www.callwheels.com
Run Date: 2/2/06 1:24 PM

TO: Reims Collectables
59 rue de l'Abbaye, null
Reims, null 51100 France

INVOICE

Attn: Paul Henriot
Sales Rep: 1337
Terms: Net 30 days

Invoice #: 10121
Account Number: 353
Date: May 07, 2003

SKU	Product Description	Price/Unit	Qty Ordered	Total Price
S50_4713	2002 Yamaha YZR M1	$74.85	44	$3,293.40
S24_2360	1982 Ducati 900 Monster	$76.88	32	$2,460.16
S32_4485	1974 Ducati 350 Mk3 Desmo	$86.74	25	$2,168.50
S12_2823	2002 Suzuki XREO	$165.68	50	$8,284.00
S10_1678	1969 Harley Davidson Ultimate Chopper	$81.35	34	$2,765.90
				$18,971.96

Send Payment and Remittance Slip to:
Steel Wheels
500 International Speedway
Daytona, FL 32114
Thank you for your business!

Account Number: 353 Reims Collectables Page 1 / 5

REMITTANCE

Invoice #: 10121
Account Number: 353

Reims Collectables
59 rue de l'Abbaye, null
Reims, null 51100 France

AMOUNT ENCLOSED:

METADATA TERMINOLOGY : DOMAIN

- Domain represents all of the associated modeled business entities

- Each domain is one view of the data
  - Servers as the highest level container and namespace
  - Can be viewed as metadata “document”

- Each solution is restricted to have at most one domain
  - Remember: solution repository can have multiple solutions
  - Domain must be published as file metadata.xmi to the solution root
  - Metadata editor only works with a domain at time

- Examples:
  - “Sales” domain to define the relationships and entities used by sales team

PHYSICAL LAYER

- Connection
  - Defines the location, user name, password, etc. for communicating database
  - Domain may contain several connections
  - Serves as a parent to other physical layer elements

- Physical Table
  - Represents an actual table in the database
  - Associated with exactly one connection

- Physical Column
  - Maps to an actual column of a table in the database
  - Associated with exactly one physical table

BUSINESS LAYER

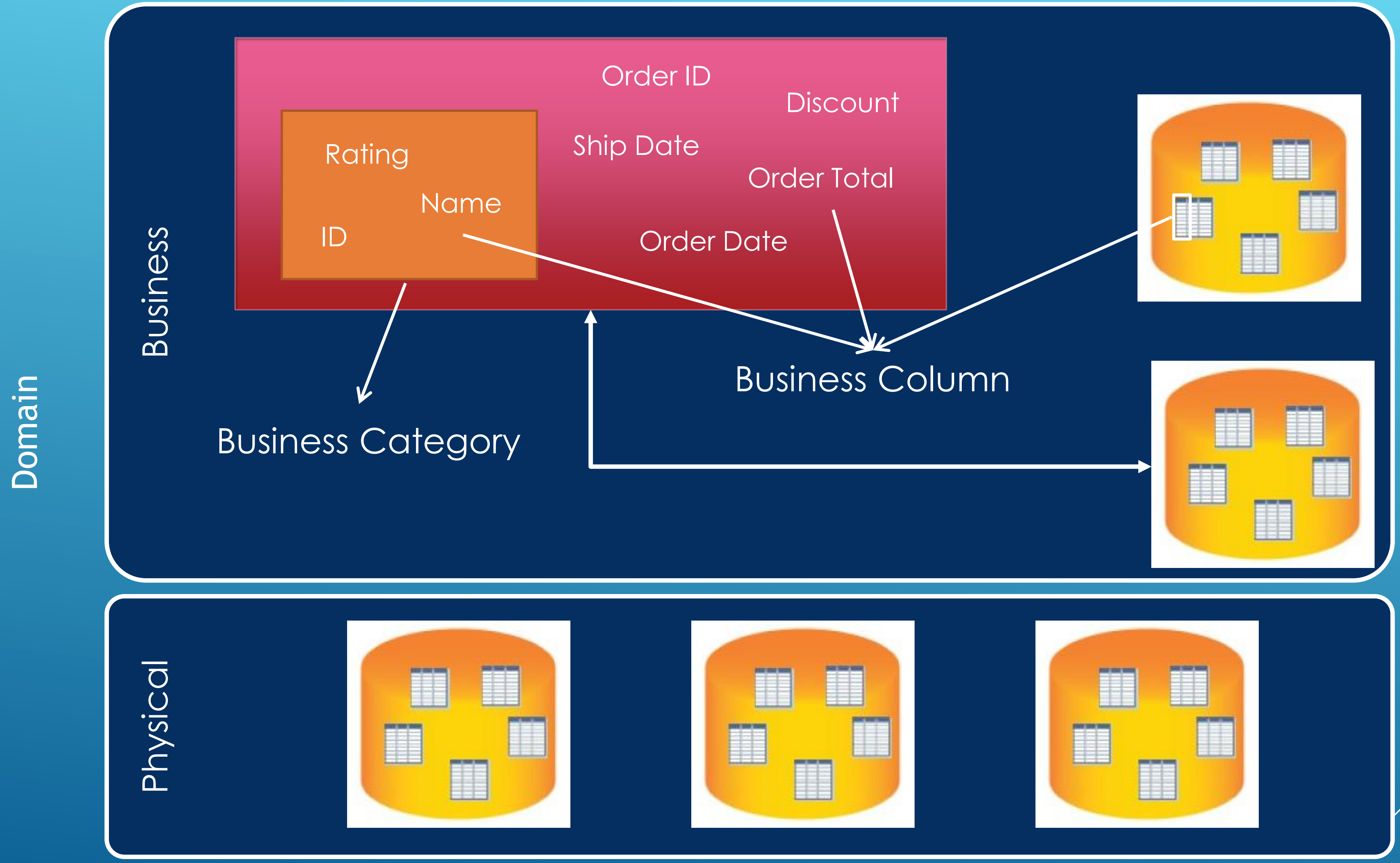
- Business layer
  - Insulates the solution from changes to the physical layer
  - Changes in the physical layer are minimized by the logical layer
  - Not every physical layer entity will be mapped in the logical layer
- Business Model
  - Logical parallel to the physical connection; mapped to a single connection
  - Contains the business tables, business columns, and business relationships
- Business Table and Business Column
  - Logical mapping of physical table and physical column
  - Minimize impact of changes in physical table\physical column

BUSINESS VIEWS

- Business View
  - Represents the business user's window on the logical level

- Business Category
  - Child of business model
  - A logical grouping of business columns
  - Members of category can come from different business tables of same model
  - Example: Customer category could contain customer name, address, customer ID, credit rating, etc. (all from different tables)

- Business View
  - Collection of business categories
  - Each business model contains a single view
  - Business user's "view" of model


Domain
Business
Order ID
Discount
Rating
Ship Date
Order Total
Name
ID
Order Date
Business Column
Business Category
Physical

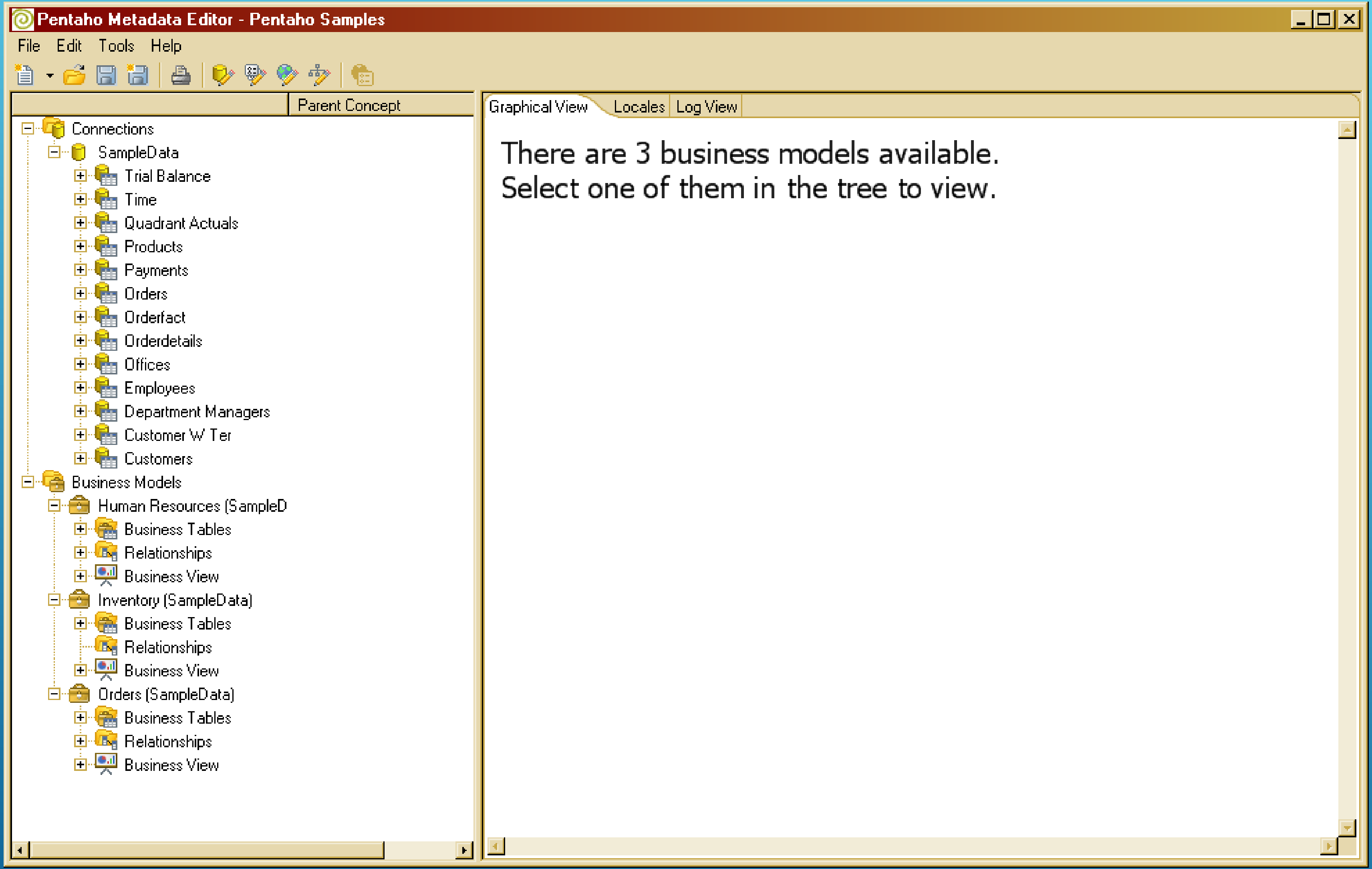
PENTAHO METADATA EDITOR

- 100% Java application
- Client application to edit and create metadata model (metadata.xmi)
- Provides a graphical user interface
- Publishes metadata to the Pentaho BI Platform\Server
- Interfaces with the Pentaho BI Platform\Server security
- Requires a JDBC connection to the data

STEPS TO CREATE METADATA MODEL

- Create a new Domain.
- Add a database connection.
- Add Physical Tables
  - Right-click on connection in navigation tree, choose **Import Tables…**
  - Right-click on existing physical table in navigation tree, choose **New Physical Table…**
- Physical Columns
  - Contained as “children” of physical table; deleted with physical table
  - Must exist in database or be calculations
  - All physical column management done in physical table property editor
- Creating and configuring a Business Model
- Creating and Managing Business Tables and Columns
- Defining relationship between the Business Tables
- Managing Business Categories
- Save and Publish to the Pentaho BA Server

COMPLETED MODEL



- Connections
  - SampleData
    - Trial Balance
    - Time
    - Quadrant Actuals
    - Products
    - Payments
    - Orders
    - Orderfact
    - Orderdetails
    - Offices
    - Employees
    - Department Managers
    - Customer W Ter
    - Customers
- Business Models
  - Human Resources (SampleD
    - Business Tables
    - Relationships
    - Business View
  - Inventory (SampleData)
    - Business Tables
    - Relationships
    - Business View
  - Orders (SampleData)
    - Business Tables
    - Relationships
    - Business View

There are 3 business models available.
Select one of them in the tree to view.