# Oracle Database 10*g*: Oracle Spatial Network Data Model

*An Oracle Technical White Paper*
*May 2005*

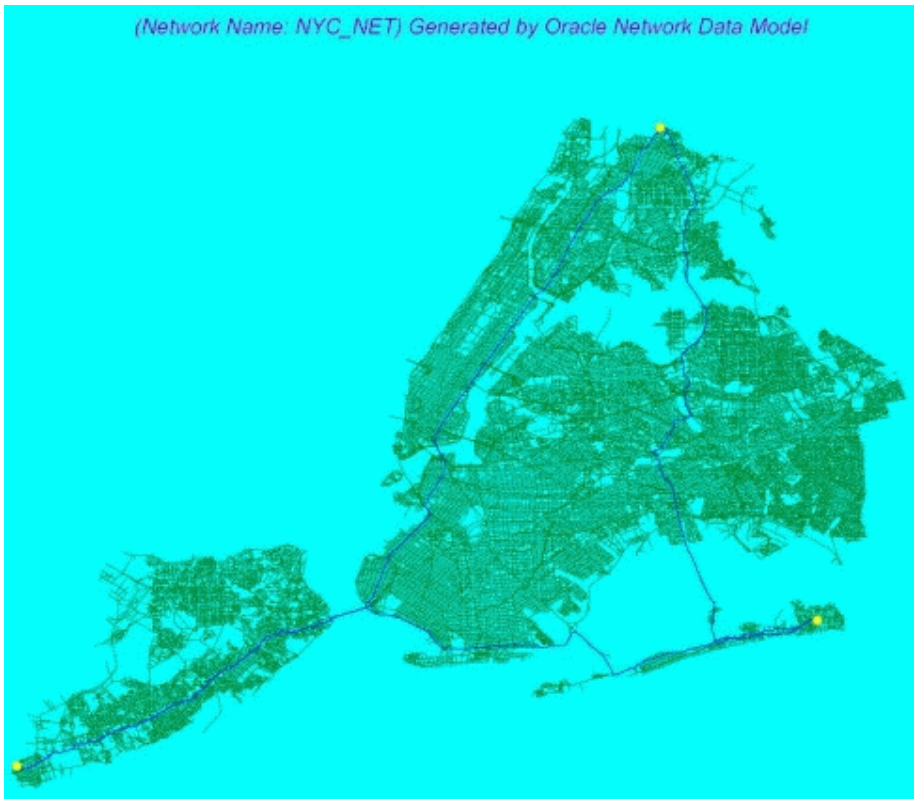
ORACLE

## Table of Contents

## INTRODUCTION

Many applications require the ability to model and analyze relationships among objects of interest. A network (or a graph) is one of such modeling representations. There are three elements in a network representation: nodes, links, and paths. The objects of interest are called nodes and the relationships between nodes are called links. A path is an ordered list of links with no repeating nodes or links. Note that only connectivity information is needed for modeling a network. In addition, the concept of cost (or weight) is introduced in links and nodes if it needs to be taken into account during analysis. Figure 1 shows a road network of New York City.

**Figure 1: New York City Road Network (60384 nodes, 151962 links. Source: NavStreets from NavTech)**




Oracle Spatial in Oracle Database 10*g* has a network data model that lets you model and analyze networks. This network data model consists of two components: a

network database schema and a Java API. The network schema contains network metadata, tables for node, link, and path metadata, and a PL/SQL package (SDO_NET). The Java API enables network representation and network analysis.

The network data model can represent both logical and spatial networks. Logical networks do not have any spatial information associated with them. Spatial networks contain spatial information that can represented using any Oracle Spatial geometry (SDO geometry, LRS geometry, or topology geometry). In addition, hierarchical relationships can be represented by organizing nodes at different hierarchy levels.

The Oracle Spatial network data model provides an open and persistent network data model, simplifies network data management and analysis, separates connectivity and application information so that applications can focus on specific application knowledge, and integrates with Oracle Spatial technology for spatial information management.

The rest of this paper discusses the approach of the network data model, network metadata and schema objects, and the network PL/SQL and Java APIs. It also includes usage information with code excerpts for network creation, loading, and analysis.

## DESIGN GOALS AND ARCHITECTURE

The network data model has the following design goals:

- Provide an open and generic network data model for network applications.

  Network data model information is stored in tables in the database. Standard SQL queries and PL/SQL functions and procedures can be issued. Only generic connectivity information is captured in the network data model, thus providing a clean separation between connectivity and application information. The network constraint mechanism enables application-related constraints to guide network analysis without knowing the application context.

- Enable spatial information support.

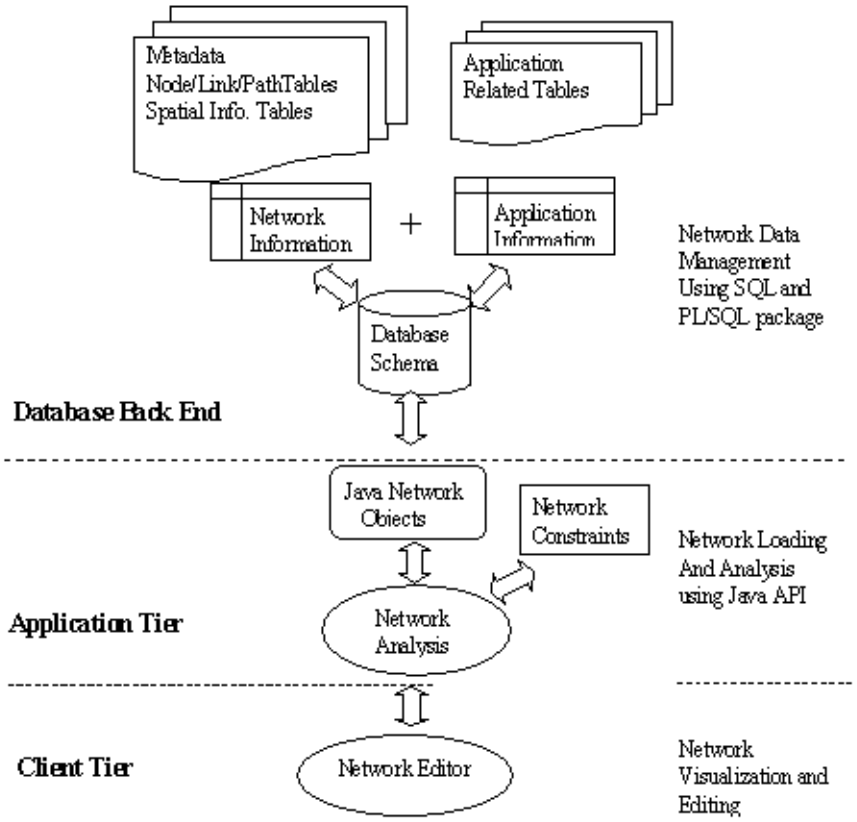
  Spatial information can be associated with network. The geometries can be store in any Oracle Spatial format: SDO geometry, LRS geometry, or topology geometry.

- Provide a 2-tier or n-tier architecture.

  Network information is stored and managed (through PL/SQL and SQL) in the relational database, and network representation, network loading, and network analysis are done in the client or application tier using the Java API.

Figure 2 shows the architecture of the network data model.

**Figure 2: Network Data Model Architecture**




## MAJOR STEPS USING THE NETWORK DATA MODEL

There are two major steps in using the network data model: network modeling and network analysis.

### Network Modeling

Network modeling transforms network applications into a network representation (nodes, links, and paths) so that network analysis can be performed. Applications extract connectivity information and maintain mapping relationships between network elements and application features.

For example, a road network consists of highways and cities, which are mapped to links and nodes, respectively. The distance or travel time can be the cost associated with each link. When a routing analysis is performed, the shortest (or fastest) path is returned in an ordered link list. The link list is mapped back to the application domain for turn-by-turn directions.

To model a network, users can take either of the following approaches:

- Create a network in the database using SQL or PL/SQL.

The network data model provides a PL/SQL package (SDO_NET) to create persistent network models in the database. Network metadata is also stored, since it provides information about the networks.

- Create a network in the client tier using the Java API.

  Networks can also be created using the Java API provided by the network data model. This API contains Java representations (interfaces and classes) of network elements, and it can be used to create Java network representations. The Java network objects are transient and can be stored in the database.

### Network Analysis

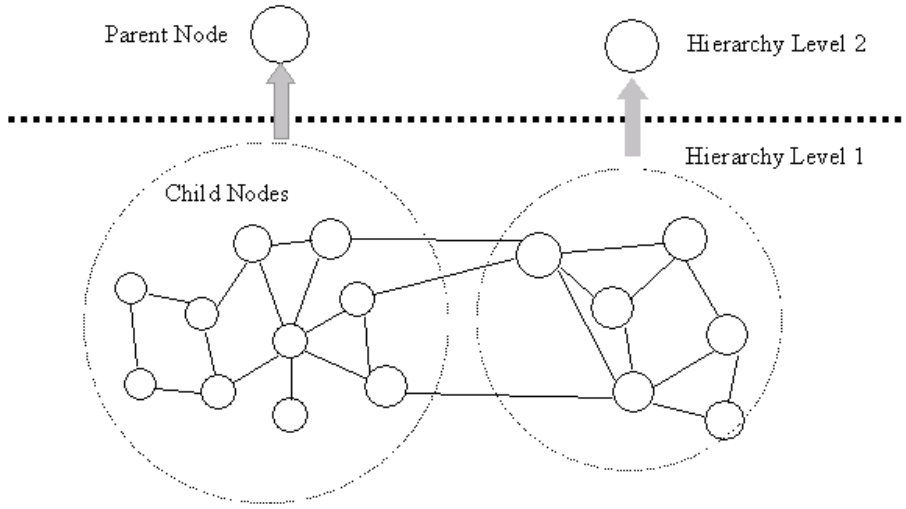
Network analysis in the network data model is conducted in the client or application tier using the Java API. When a network is loaded from the database, a Java network object is created and is ready for analysis; however, you should validate a network (metadata and referential integrity) before analyzing it.

## NETWORK MODELING

The network data model can be used to model logical networks (without spatial information) and spatial networks (with spatial information). For spatial networks, the following kinds of Oracle Spatial types are supported: SDO geometry, LRS geometry, and topology geometry.

Both spatial and logical networks can be modeled with hierarchical relationships (parent-child), as shown in Figure 3.

**Figure 3: Hierarchical Relationships in a Network**




## METADATA AND SCHEMA

Network metadata consists of information about networks. For each network that you create, you must insert information into the

USER_SDO_NETWORK_METADATA view. This view includes columns for the following information, although columns that do not apply to a specific network can contain null values:

- NETWORK (name of the network)
- NETWORK_CATEGORY ('LOGICAL' or 'SPATIAL')
- GEOMETRY_TYPE ('SDO_GEOMETRY', 'LRS_GEOMETRY', or 'TOPO_GEOMETRY' for a spatial network; NULL for a logical network)
- NODE_TABLE_NAME
- NODE_GEOM_COLUMN
- LINK_TABLE_NAME
- LINK_GEOM_COLUMN
- LRS_TABLE_NAME
- LRS_GEOM_COLUMN
- PATH_TABLE_NAME
- PATH_GEOM_COLUMN
- PATH_LINK_TABLE_NAME
- LINK_COST_COLUMN
- NODE_COST_COLUMN
- LINK_DIRECTION ('DIRECTED' or 'UNDIRECTED')
- NO_OF_HIERARCHY_LEVELS

In addition, for each network you must create and populate the following tables to contain information about nodes, links, and paths:

- Node table
- Link table
- Path table (only if the network contains paths)
- Paths and links table (only if a path table is created; contains a row for each link in each path in the network)

## NETWORK JAVA REPRESENTATION AND API

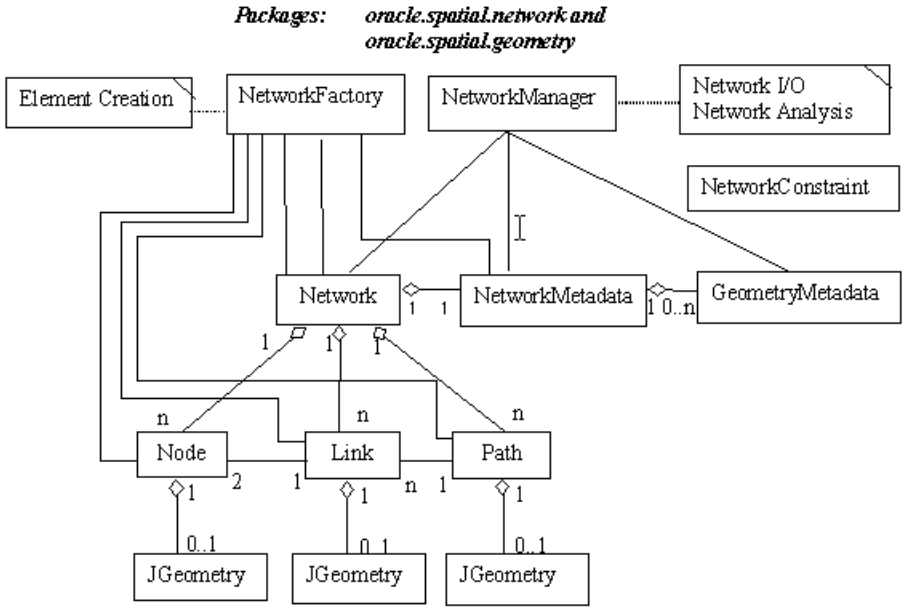
The network data model provides a Java API for network representation and analysis. The Java client interface for the network data model consists of the following classes and interfaces:

- NetworkMetadata: network metadata
- GeometryMetadata: geometry metadata for spatial network

- Network: network representation
- Node: network node
- Link: network link
- Path: network path
- NetworkDataException: network data model exception
- JGeometry: Java SDO Geometry
- NetworkConstraint: Search constraint for network analysis

Figure 4 shows the relationship between the main classes and interfaces.

**Figure 4: Java Classes and Interfaces for Network Data Model**




Two java classes are used for network management and network analysis:

- NetworkFactory: network creation, network elements creation
- NetworkManager: network loading, storing, and analysis

## NETWORK CREATION USING SQL AND PL/SQL

The network data model provides PL/SQL procedures (package SDO_NET) to simplify network creation and management. You can create network tables with predefined table names, column names, and default settings. The network metadata views are automatically updated when a network is created. The default values for network table names, column names, and metadata are as follows:

- Node table name: <network>_NODE$
- Link table name: <network>_LINK$

- Path table name: <network>_PATH$
- Path Link table name: <network>_PLINK$
- Node geometry column: GEOMETRY
- Link geometry column: GEOMETRY
- Path geometry column: GEOMETRY
- Link cost column: COST
- Node cost column: null (node without cost) or COST (node with cost)

This section describes the main steps for creating logical and spatial networks. However, for detailed instructions and explanations, see the Oracle Spatial Topology and Network Data Models manual.

### Creating a Logical Network

In the network metadata for a logical network, the NETWORK_CATEGORY is LOGICAL and the GEOMETRY_TYPE is null.

To create a logical network, follow these main steps:

1. Create network tables and insert network metadata. For example:

   ```
   EXEC SDO_NET.CREATE_LOGICAL_NETWORK(
     'LOG_NET', -- network name
       1, -- no of hierarchy level
       true, -- directed link?
       false, -- no with cost?
   );
   ```

2. Populate the node and link tables.
3. Validate the network. For example:

   ```
   SDO_NET.VALIDATE_NETWORK('LOG_NET');
   ```

### Creating a Spatial Network

A spatial network contains geometries of one of the following three types:

- SDO geometries: objects of type SDO_GEOMETRY that are not LRS geometries (that is, do not contain a measure dimension)
- LRS geometries: objects of type SDO_GEOMETRY that are LRS geometries (that is, do contain a measure dimension)
- Topology geometries: objects of type SDO_TOPO_GEOMETRY

#### Creating an SDO Geometry Network

In the network metadata for a spatial network with SDO geometries, the NETWORK_CATEGORY is SPATIAL and the GEOMETRY_TYPE is SDO_GEOMETRY.

To create a spatial network with SDO geometries, follow these main steps:

1. Create network tables and insert network metadata. For example:

   ```
   EXEC SDO_NET.CREATE_SDO_NETWORK(
     'SDO_NET', -- network name
     1, -- no of hierarchy level
     true, -- directed link?
     false, -- node with cost?
   );
   ```

2. Populate the node and link tables.
3. Insert geometry metadata for the node and link tables, and the path and paths and links tables if paths will be used.
4. Validate the network. For example:

   ```
   SDO_NET.VALIDATE_NETWORK('SDO_NET');
   ```

#### Creating an LRS Geometry Network

In the network metadata for a spatial network with LRS geometries, the NETWORK_CATEGORY is SPATIAL and the GEOMETRY_TYPE is LRS_GEOMETRY.

To create a spatial network with LRS geometries, follow these main steps:

1. Create network tables and insert network metadata. For example:

   ```
   EXEC SDO_NET.CREATE_LRS_NETWORK(
     'LRS_NET', -- network name
     'LRS_GEOM_TABLE', -- LRS geometry table name
     'LRS_GEOM_COLUMN', -- LRS geometry column name
     1, -- no of hierarchy level
     true, -- directed link?
     false, -- node with cost?
     );
   ```

2. Populate the node and link tables.

3. Insert geometry metadata for the node and link tables, and the path and paths and links tables if paths will be used.
4. Validate the network. For example:

   SDO_NET.VALIDATE_NETWORK('LRS_NET');

#### Creating a Topology Geometry Network

In the network metadata for a spatial network with topology geometries, the NETWORK_CATEGORY is SPATIAL and the GEOMETRY_TYPE is TOPO_GEOMETRY.

To create a spatial network with topology geometries, follow these main steps:

1. Create network tables and insert network metadata. For example:

   EXEC SDO_NET.CREATE_TOPO_NETWORK(

   'TOPO_NET', -- network name

   1, -- no of hierarchy level

   true, -- directed link?

   false, -- no with cost?

   );
2. Populate the node and link tables.
3. Insert geometry metadata for the node and link tables, and the path and paths and links tables if paths will be used.
4. Validate the network. For example:

   SDO_NET.VALIDATE_NETWORK('TOPO_NET');

## NETWORK CREATION USING THE JAVA API

As an alternative to using PL/SQL and SQL, you can use the network data model Java API to create a network. There are java classes and interfaces that represent network and network elements at the application or client tier. To create a Java network, create an empty network object with network metadata, and add nodes, links, and optionally paths. Once the Java network object is created, you can perform network analysis on it. You can also store the Java network object in the database.

The following code excerpt creates a network, performs some analysis, and stores the network in the database:

…

// create an empty logical network

Network logicalNetwork

```
= NetworkFactory.createLogicalNetwork(
"LOGICAL_NET",
  1, // no of hierarchy levels
  true // directed link?
  );
// create nodes and add them to the network
logicalNetwork.addNode(NetworkFactory.createLogicalNode(
1, // node ID
"N1" // Node Name
);
logicalNetwork.addNode(NetworkFactory.createLogicalNode(
2, // node ID
"N2" // Node Name
);
 …
 // create links and add them to the network
 logicalNetwork.addLink(NetworkFactory.createLogicalLink(
1, // link ID
   "L1", // link name
   logicalNetwork.getNode(1),
   logicalNetwork.getNode(2),
…
 // find the shortest path
 Path path = logicalNetwork.shortestPath(logicalNetwork,1,2);
 path.setID(1); // set Path ID
 //add the path to network
 logicalNetwork.addPath(path);
 // save the network to database
NetworkManager.writeNetwork(con,logicalNetwork);
```

## NETWORK ANALYSIS USING THE JAVA API

Once a network is stored in the database, the Java API can load the network and perform network analysis. The network data model currently provides the following analysis capabilities:

- Tracing:

  Find all nodes which can be reached from a given node

  Find all nodes which can reach a given node

- Shortest Path:

  Find the shortest path between a start node and a goal node

- Minimum Cost Spanning Tree:

  Find the minimum cost spanning tree which connects all nodes

- Within Cost:

  Find all nodes which are within a given cost of a given node

- Nearest Neighbors:

  Find N closest nodes of a given node based on cost

The following code excerpt shows some network analysis:

```
try {
  // load a network named "LOG_NET" from database
Network network = NetworkManager.readNetwork(con,"LOG_NET");
…
// find the shortest path between startNodeID and EndNodeID
   Path path = NetworkManager.shortestPath(network,startNodeID,endNodeID);
  …
  // check if startNodeID can be reached by EndNodeID
  if ( NetworkManager.isReachable(network,startNodeID,endNodeID) ) {
   …
  }
  // find 10 nearest neighbors from startNodeID
  Path [] pathArray =
Networkmanager.nearestNeighbors(network,startNodeID,10);
  …
  // find all nodes which are within cost 20 from startNodeID
```

```
  pathArray = NetworkManager.withinCost(network,startNodeID,20.0);
  // find all links in the minimum cost spanning tree of the network
Link [] linkArray = NetworkManager.mcstLinkArray(network);
  } catch (NetworkDataException exp) {
}
```

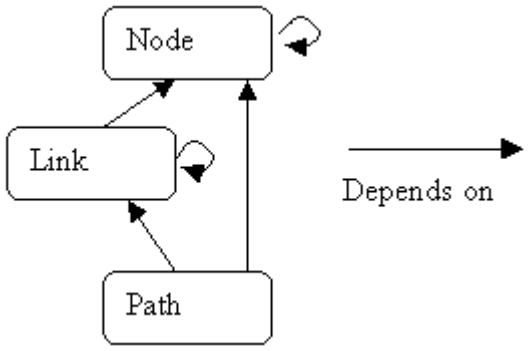
## REFERENTIAL INTEGRITY OF NETWORK DATA

To ensure the referential integrity of network data, you can create referential constraints to be applied to network tables. You can create, enable, and disable referential integrity checking on a network. If referential integrity checking is enabled for a network, the following dependencies are enforced:

- Node/link dependency: A link depends on its start node and end node. If a start or end node is deleted, the link is deleted. No link can be added if the nodes it depends on do not exist.
- Path/link dependency: A path depends on its links. If any link of the path is deleted, the path is deleted as well. No path can be added if the links it depends on do not exist.
- Hierarchical dependency: Elements at a higher hierarchy level cannot exist if their dependent elements do not exist.

Figure 5 shows the dependency relationships of a network. You can validate a network to check network metadata, schema, and referential integrity of a network and its elements.

**Figure 5: Network Element Dependency**




## NETWORK CONSTRAINTS

Network constraints are useful filters for network analysis. Network constraints can quickly limit the search to desired targets at a very early stage. Network analysis is usually conducted in a very specific (constrained) manner to limit the number of possible solutions.

The network data model lets applications control network analysis without knowing how the analysis is conducted. This feature complements the separation of connectivity information from application information. The network data model provides two kinds of network constraints: system constraints and user constraints.

System constraints are common constraints for network analysis. They include the following:

- MBR constraint: Nodes must be within given MBR in order to be considered as valid candidate.
- Path cost constraint: Paths cannot have a path cost greater than the specified cost.
- Path depth constraint (number of links): Paths cannot have a path length greater than the specified number.
- Must-avoid nodes and links: The specified nodes and links cannot be included in paths.

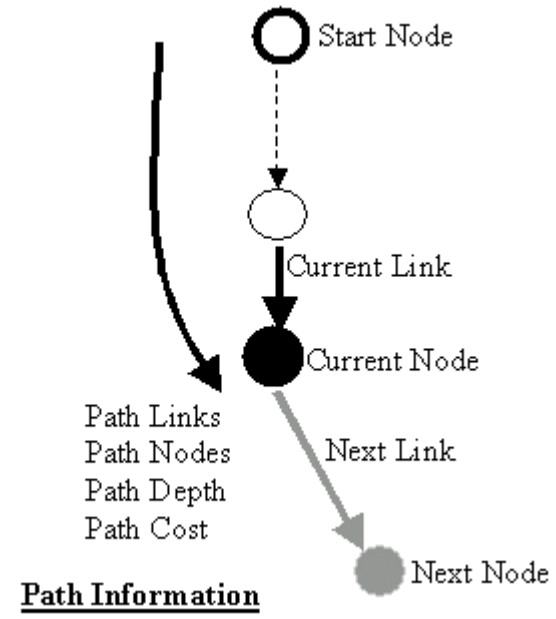
User constraints, on the other hand, are application-dependent constraints that can be implemented through the NetworkConstraint interface. Applications are responsible for providing the implementation and guiding network analysis if such constraints exist.

There are two methods that need to be implemented in NetworkConstraint:

- boolean requiresPathLinks(), to indicate whether or not all path links and nodes are needed for implementing the constraint.
- boolean isSatisfied(AnalysisInfo info), to check if the constraint is satisfied. The AnalysisInfo is passed to applications, and it contains the following information about the current search: start node, current node, next node, current link, next link, current depth (number of links), current cost, and path links and nodes (if requiresPathLinks() is true).

Figure 6 shows the analysis information for network constraints.

**Figure 6: Analysis Information for Network Constraints**

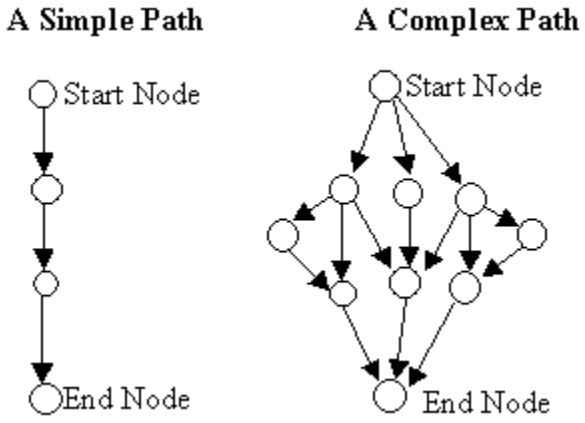



## PATH REPRESENTATION

A simple path is an ordered list of links with no repeating nodes. It is usually the result of a path computation. In some applications, however, a path is an abstract representation of a set of simple paths with a start node and end node. This kind of path is called a complex path in the network data model. Complex paths can be used in path computation to identify a set of specific simple paths, such as the shortest path or set of paths whose length is smaller than a given length threshold. The links of a complex path are not required in the paths and links table, because the network, start node, and end node provide enough information for path analysis.

Figure 7 shows simple and complex paths.

**Figure 7: Simple Path and Complex Path**



## REQUIREMENTS

The network data model is included with Oracle Spatial in Oracle Database 10g. Specifically, the network data model requires the SDO_NET PL/SQL package, the network model Java API (package: oracle.spatial.network), and the Oracle SDO Geometry Java API (package: oracle.spatial.geometry).

The Oracle JDBC driver is also required for the database connection

## NEW FEATURES IN THE ORACLE SPATIAL 10*G* NETWORK DATA MODEL

With Oracle Spatial 10*g* Release 2, the network data model provides the following features:

### Network Modeling: Link Direction

The directionality of a link can be further specified at the link level. Unlike the network directionality that determines the directions of all links, a directed network can have links that are directed or bi-directed. A BIDIRECTED column can be added to the link table and used to indicate if a directed link is bi-directed. This modeling enhancement will reduce the storage requirement for directed networks with non-homogeneous link directions (unidirectional and bidirectional).

### Network Analysis: Maximum Flow Analysis

The maximum flow analysis function is provided for a single source and single sink flow network. Each link in a flow network has a flow capacity associated with it. The goal of this function is to find the maximum allowable flow that can flow from the source node to the sink node. This type of analysis is commonly seen in communication or logistics network planning.

### PL/SQL Wrapper Package

Prior to Oracle Spatial 10*g* Release 2, only the network data model Java API could be used for network editing and analysis. Now, a PL/SQL wrapper package is also provided that helps users edit and analyze networks in PL/SQL. This wrapper package provides nearly equivalent functionality as the Java API. It is done through database Java stored procedures and Java virtual machine in Oracle.

## CONCLUSION

The Oracle Spatial network data model, available with Oracle Spatial 10*g*, provides the capabilities for modeling and analyzing network applications. The network data model is an open and generic network data model that represents networks in object-relational form in the database and as Java objects in client or application tier. The 2-tier or n-tier architecture takes advantage of database and Java capabilities. The main goals of the network data model are to cleanly separate application logic and attributes from generic network information, and to simplify network management and analysis so that network applications can focus on their

application knowledge. The network data model is tightly integrated with Oracle Spatial, to extend the spatial information management capabilities of Spatial and make it a powerful modeling and analysis platform for network applications.

## REFERENCES

For technical usage and reference information, see the following manuals in the Oracle Database 10*g* documentation set:

- *Oracle Spatial Topology and Network Data Models*
- *Oracle Spatial User's Guide and Reference*

**Oracle Spatial Network Data Model: An Oracle Technical White Paper**
**May 2005**
**Author: Jack Chenghua Wang**

**Oracle Corporation**
**World Headquarters**
**500 Oracle Parkway**
**Redwood Shores, CA 94065**
**U.S.A.**

**Worldwide Inquiries:**
**Phone: +1.650.506.7000**
**Fax: +1.650.506.7200**
**oracle.com**