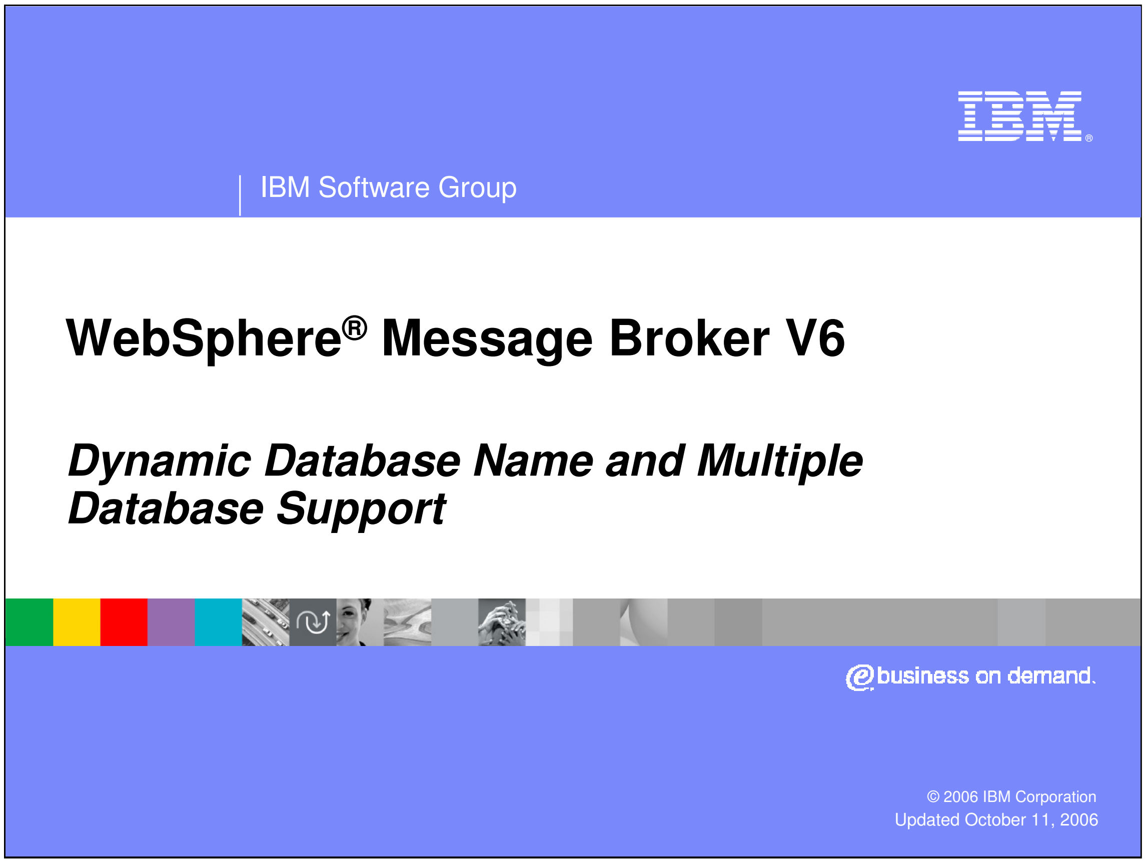


IBM Software Group

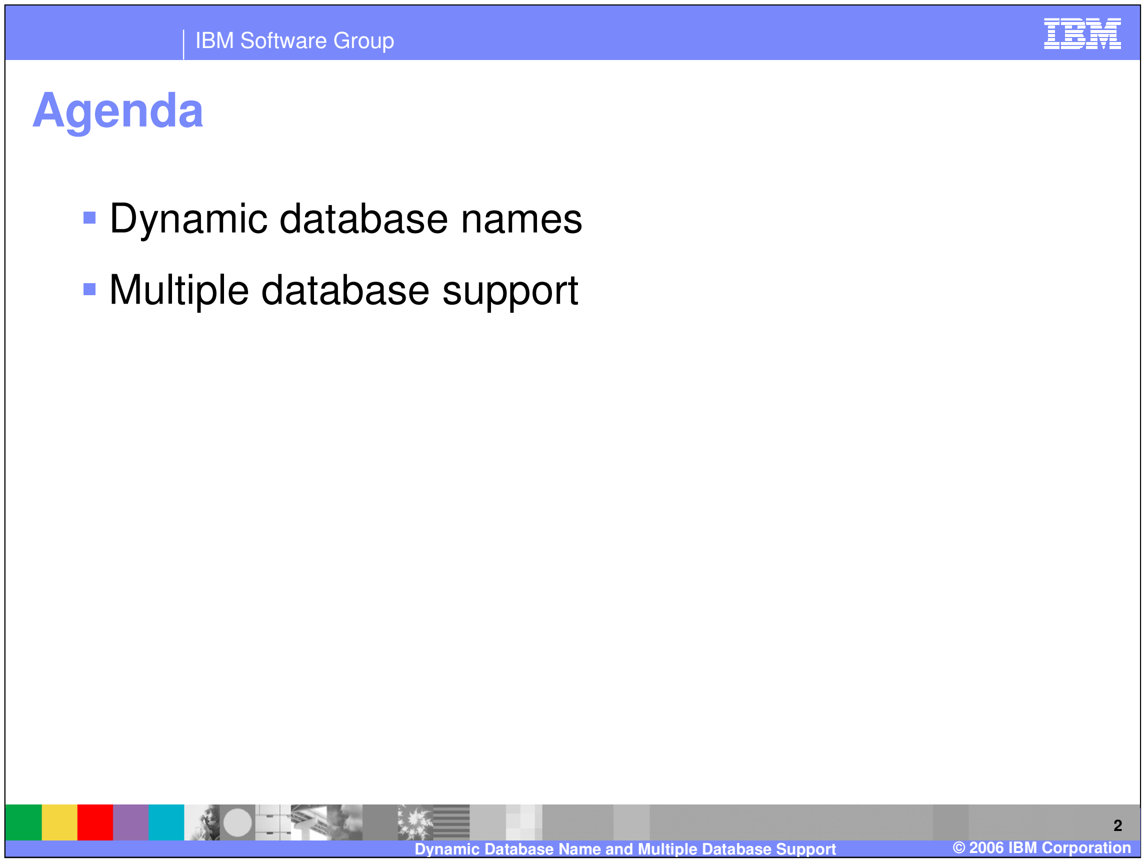
WebSphere® Message Broker V6

Dynamic Database Name and Multiple Database Support

@business on demand.

Updated October 11, 2006

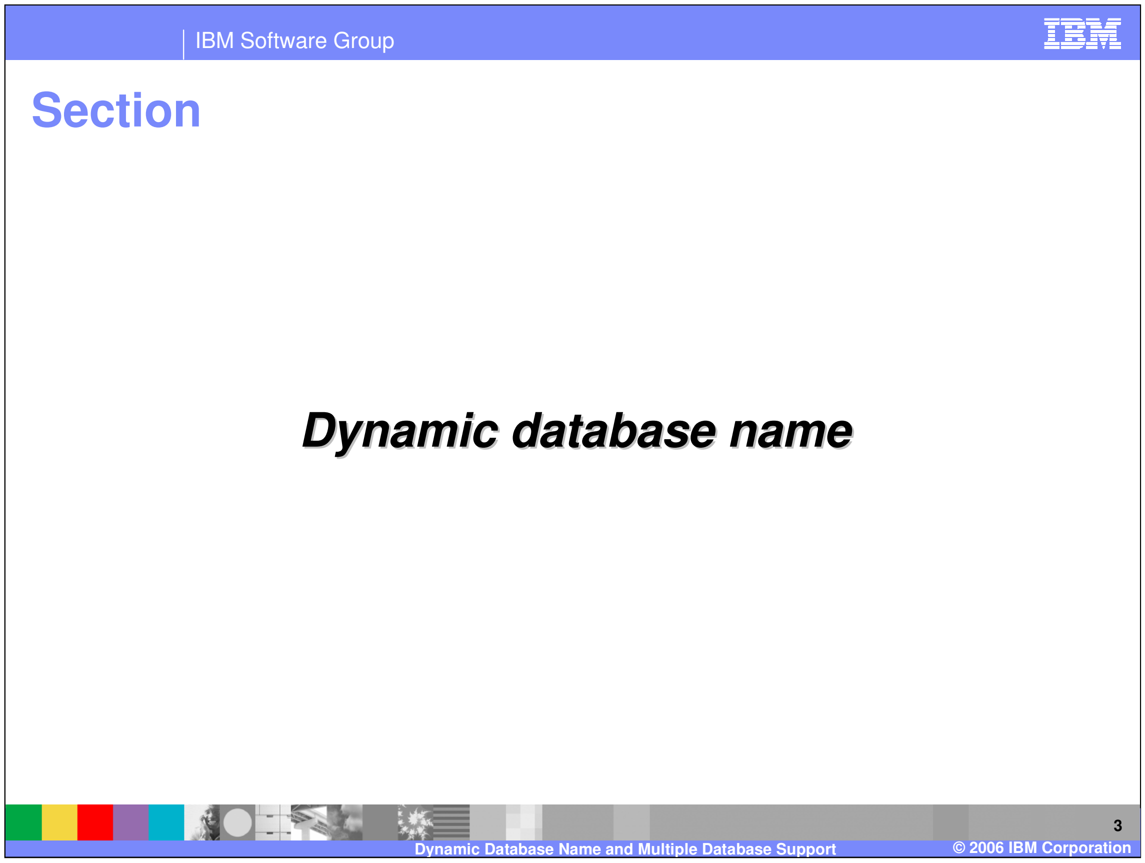
This presentation discusses Dynamic Database Name and Multiple Database Support in WebSphere Message Broker V6.



Agenda

- Dynamic database names
- Multiple database support

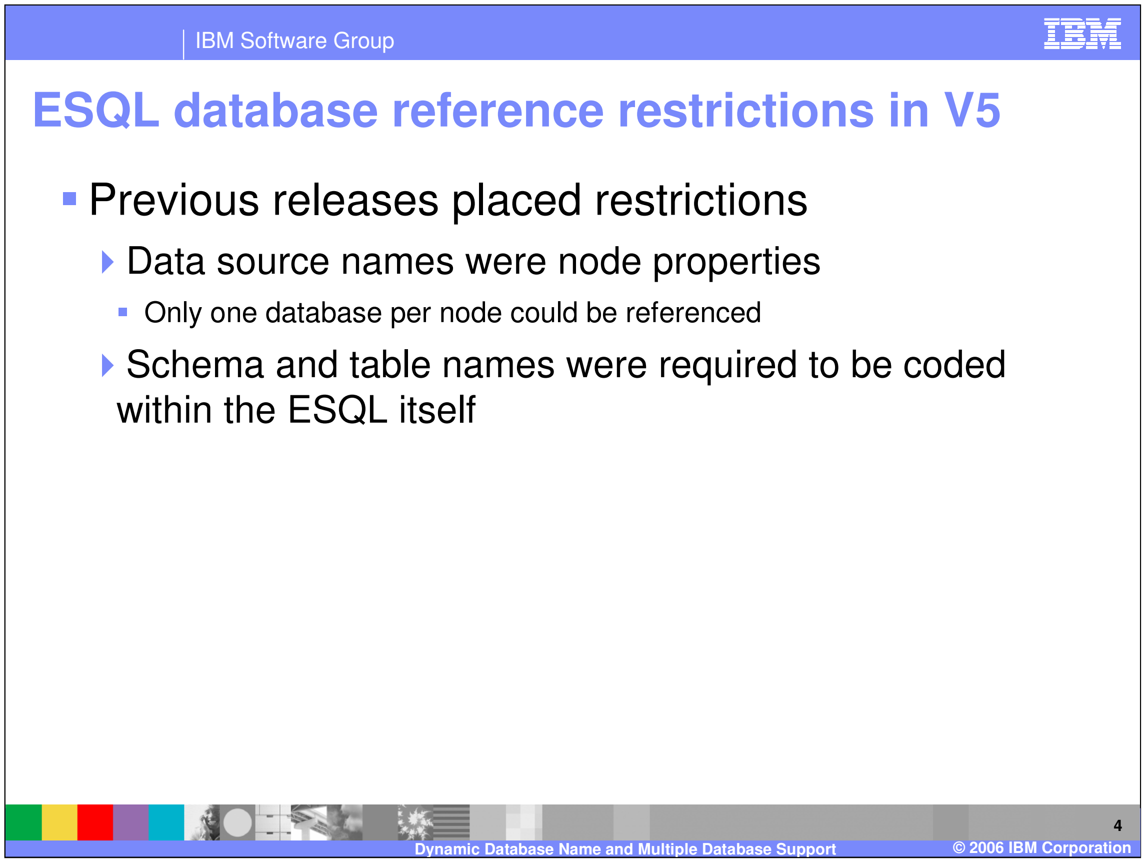
The agenda for this presentation covers the new support for Dynamic Database Names and Multiple Database Support in WebSphere Message Broker V6 for ESQL nodes.



Section

Dynamic database name

This section provides detailed information about the Dynamic Database Name support in V6.


IBM Software Group

ESQL database reference restrictions in V5

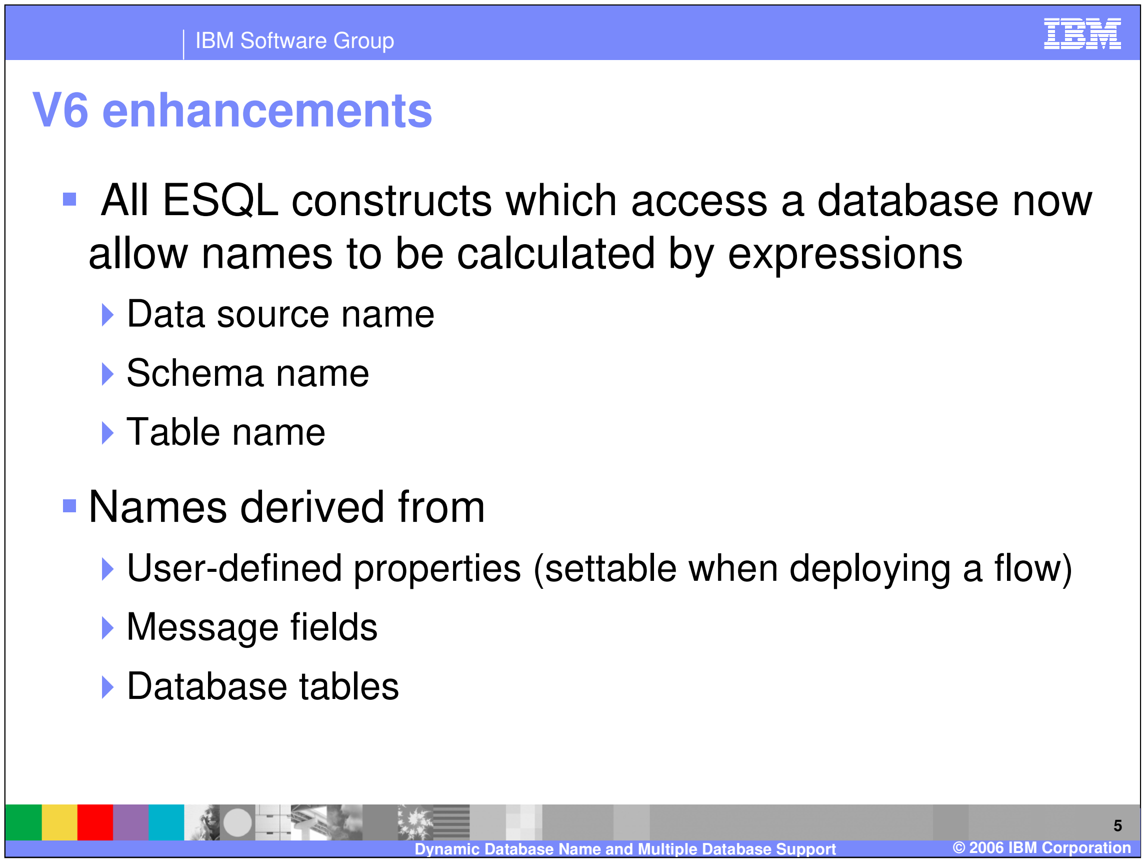
- Previous releases placed restrictions
  - Data source names were node properties
    - Only one database per node could be referenced
  - Schema and table names were required to be coded within the ESQL itself

4

Dynamic Database Name and Multiple Database Support

© 2006 IBM Corporation

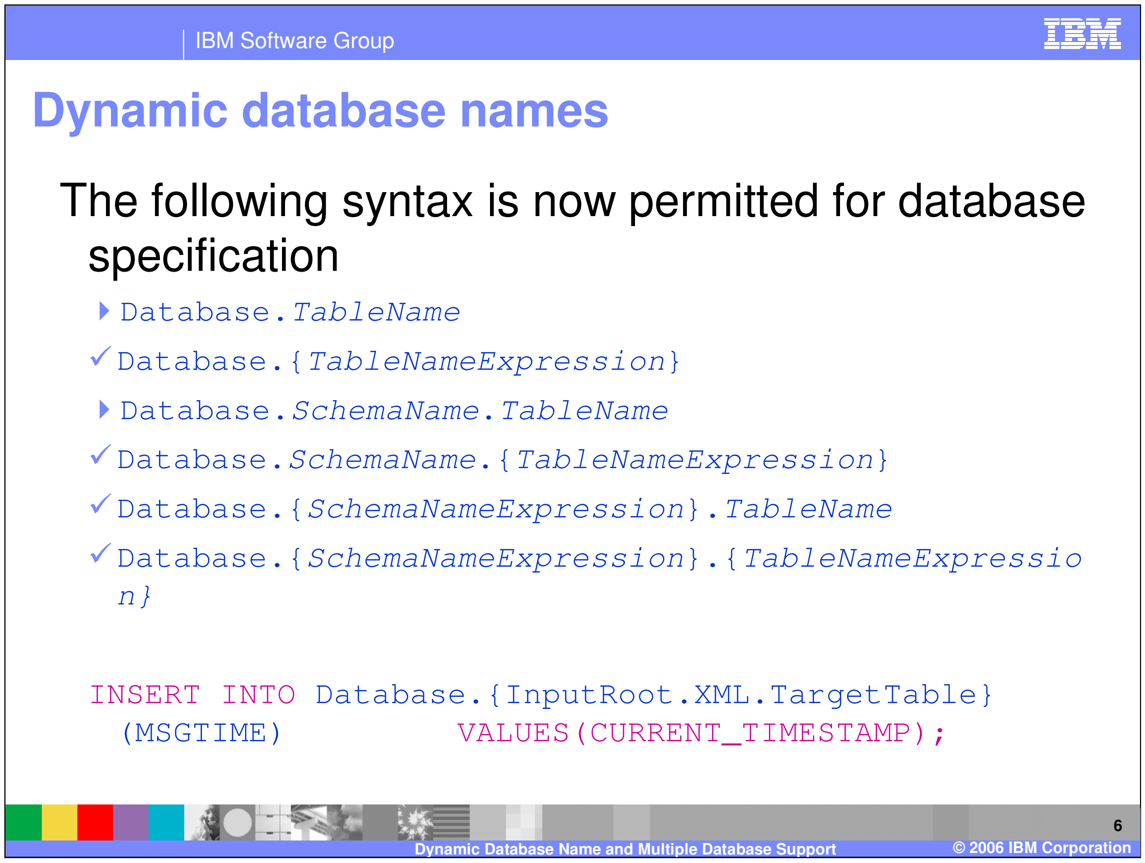
In previous releases of WebSphere Message Broker, the data source name was a node property. This results in the restriction that only one database could be referenced per node. Also, the schema name and table names were required to be hard-coded within the ESQL. Although expression statements were allowed in ESQL, expressions could not be used in database references.



V6 enhancements

- All ESQL constructs which access a database now allow names to be calculated by expressions
  - Data source name
  - Schema name
  - Table name
- Names derived from
  - User-defined properties (settable when deploying a flow)
  - Message fields
  - Database tables

In WebSphere Message Broker V6, all ESQL constructs which access a database now allow the data source, schema, and table names to be calculated by expressions. These names can be derived from user defined properties, settable when deploying a flow, from message fields, from database tables, or from any other source.



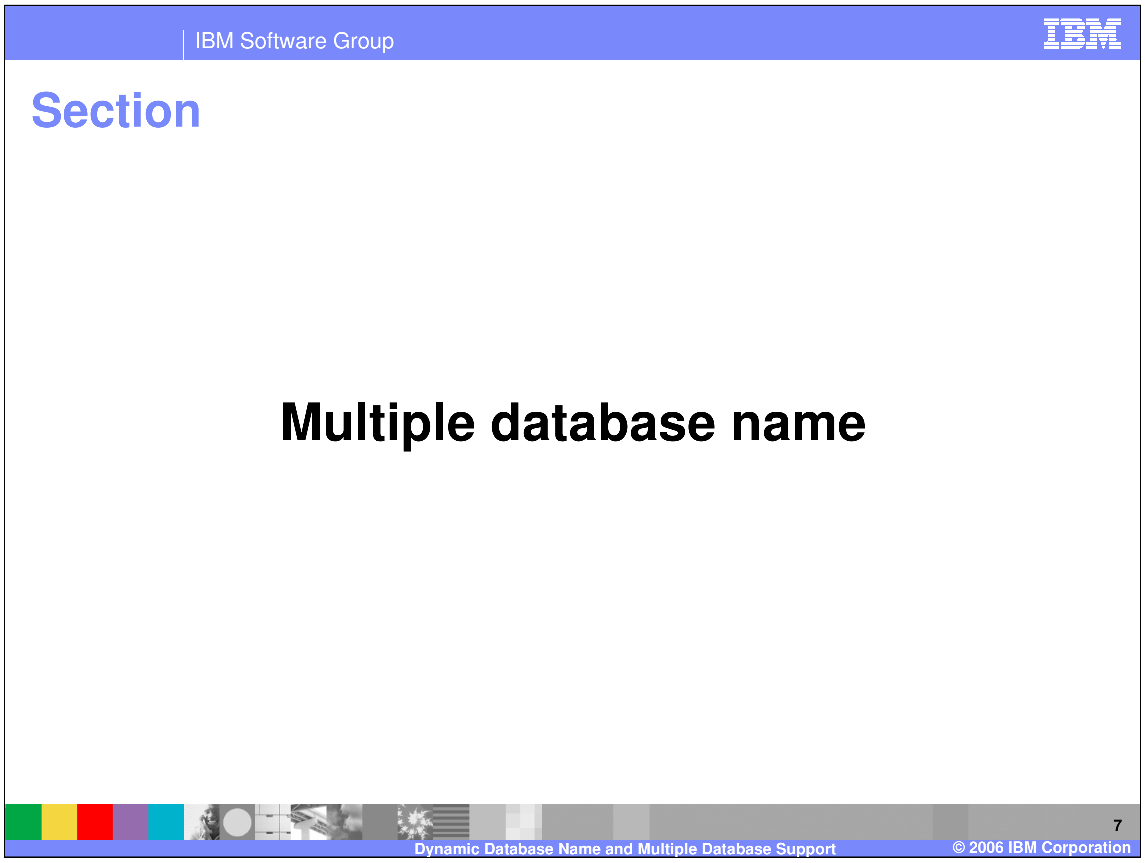
Dynamic database names

The following syntax is now permitted for database specification

- Database.*TableName*
- ✓ Database.{*TableNameExpression*}
- Database.*SchemaName*.*TableName*
- ✓ Database.*SchemaName*.{*TableNameExpression*}
- ✓ Database.{*SchemaNameExpression*}.*TableName*
- ✓ Database.{*SchemaNameExpression*}.{*TableNameExpression*}

```
INSERT INTO Database.{InputRoot.XML.TargetTable}
  (MSGTIME)         VALUES(CURRENT_TIMESTAMP);
```

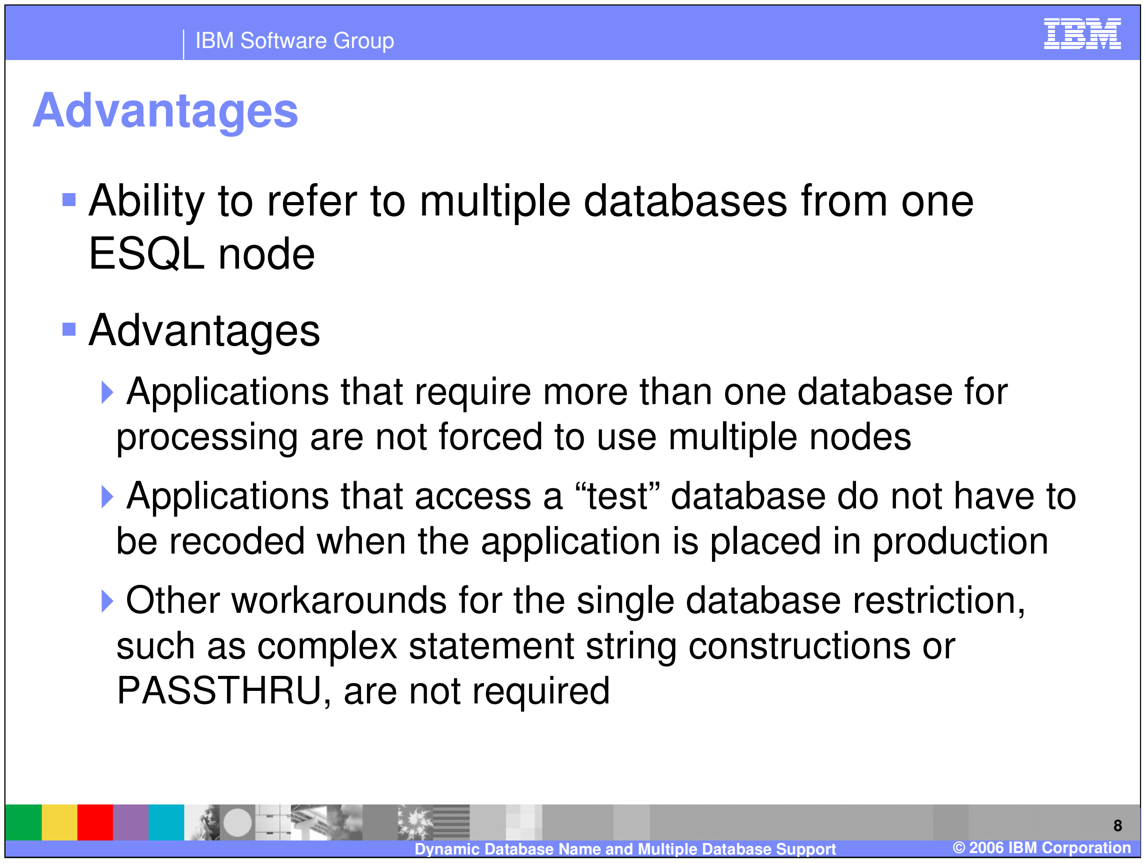
Here is a summary of the syntax that is now permitted in V6. Note that the keyword "Database" indicates a query from an external database, rather than a message tree query. If the data source name is not specified, it defaults to the data source name defined at the node level.



Section

Multiple database name

This section discusses Multiple Database Name support in V6.



Advantages

- Ability to refer to multiple databases from one ESQL node
- Advantages
  - Applications that require more than one database for processing are not forced to use multiple nodes
  - Applications that access a “test” database do not have to be recoded when the application is placed in production
  - Other workarounds for the single database restriction, such as complex statement string constructions or PASSTHRU, are not required

In previous releases, you could not reference more than one database within a node. In V6, you can now use the ESQL expression enhancements for database references to access more than one database within one ESQL node. Several advantages of accessing more than one database in a node are immediately apparent. Applications that require more than one database for processing are not forced to use multiple nodes. Applications that access a “test” database do not have to be re-coded when the application is placed in production. And other workarounds for the “single database” access restrictions in previous releases – such as complex statement string constructions or PASSTHRU – are no longer required.

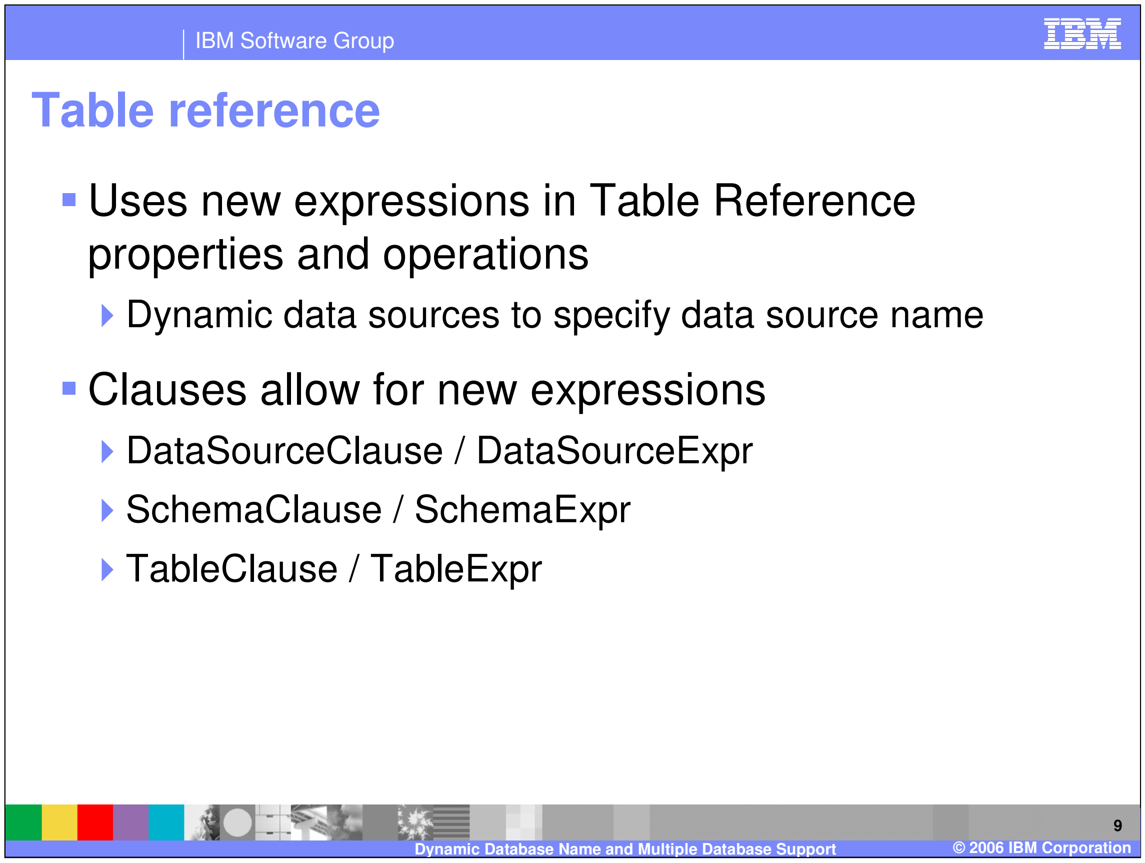

IBM Software Group

Table reference

- Uses new expressions in Table Reference properties and operations
  - Dynamic data sources to specify data source name
- Clauses allow for new expressions
  - DataSourceClause / DataSourceExpr
  - SchemaClause / SchemaExpr
  - TableClause / TableExpr

9

Dynamic Database Name and Multiple Database Support © 2006 IBM Corporation

The multiple database support requires the use of the new table reference property and operations. A table reference is simply a field reference in which the first part of the name is literally "Database". A dynamic data source is used to specify the data source name. You can use the new expression syntax within the clauses to reference data source, schema, and table.

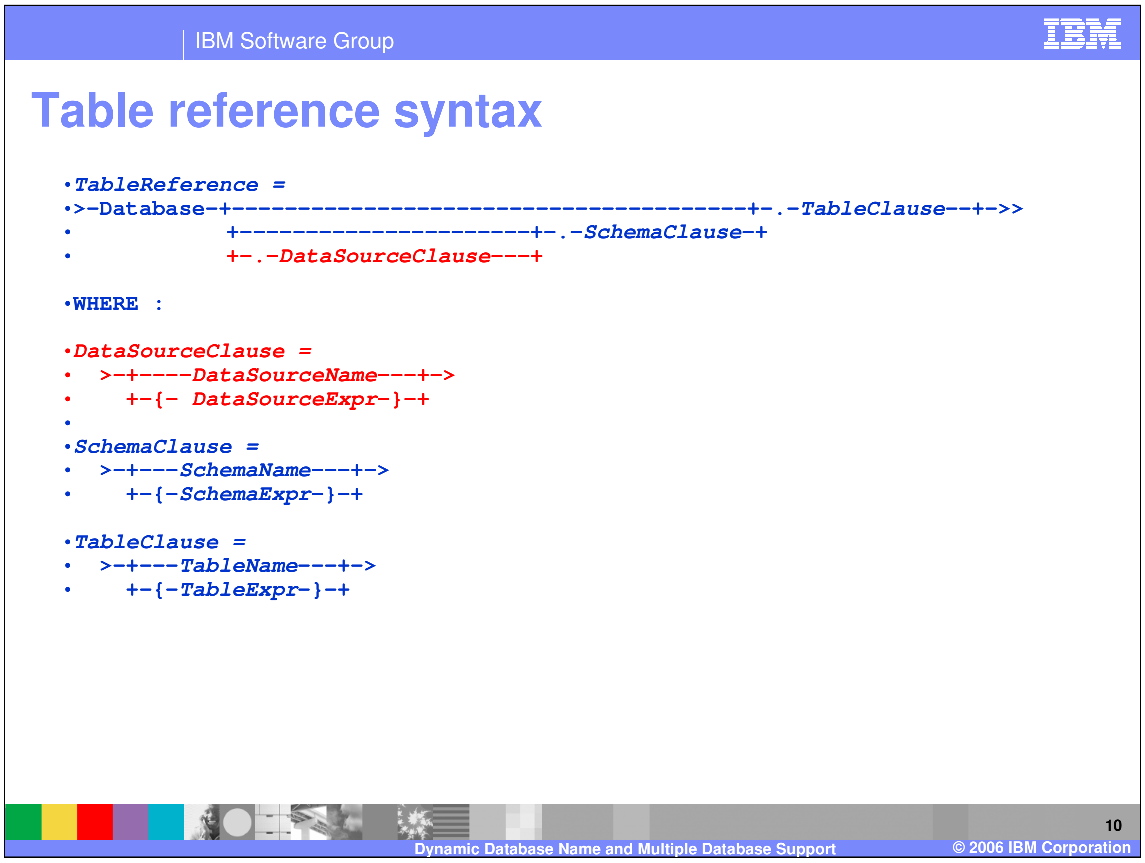

IBM Software Group

Table reference syntax

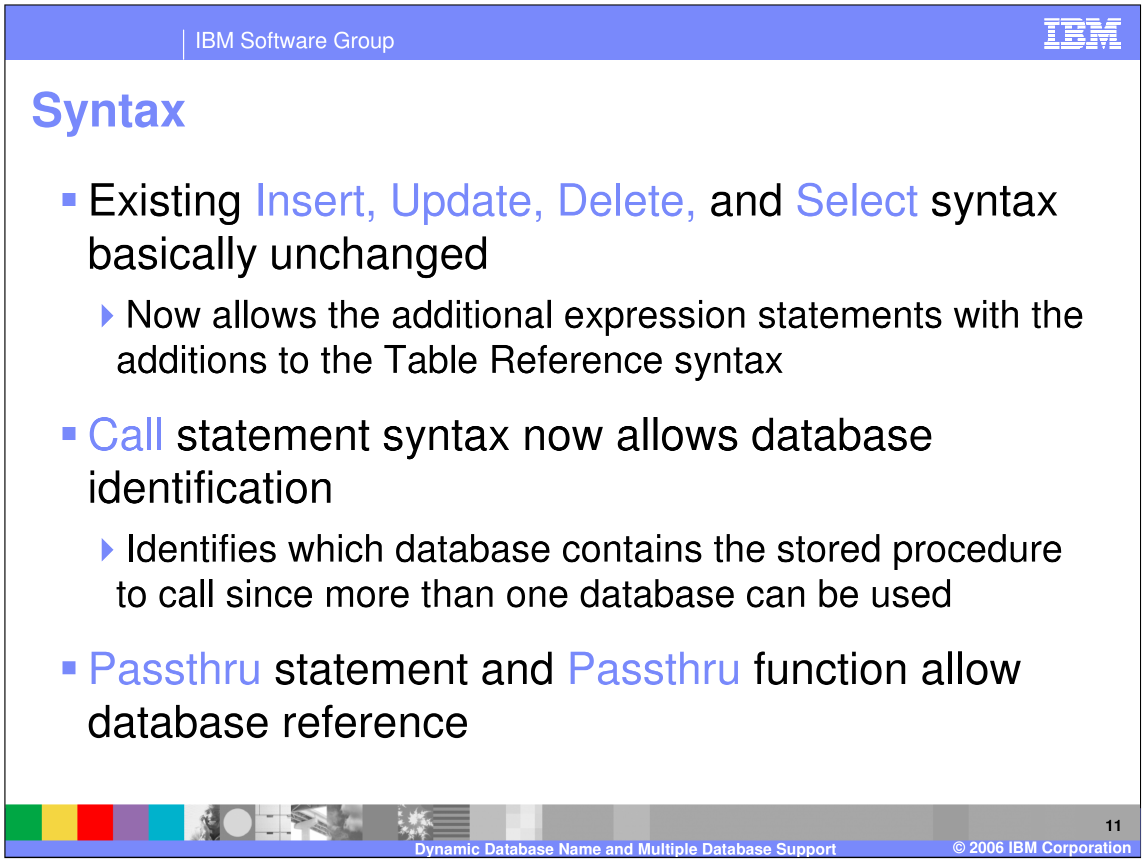
```
•TableReference =
•>-Database-+----------------------------------------+-.-TableClause--+->>
•           +-----------------------+-.-SchemaClause-+
•           +-.-DataSourceClause---+

•WHERE :

•DataSourceClause =
•  >-+----DataSourceName---+->
•    +-{- DataSourceExpr-}-+
•
•SchemaClause =
•  >-+---SchemaName---+->
•    +-{-SchemaExpr-}-+

•TableClause =
•  >-+---TableName---+->
•    +-{-TableExpr-}-+
```

10

Dynamic Database Name and Multiple Database Support

© 2006 IBM Corporation

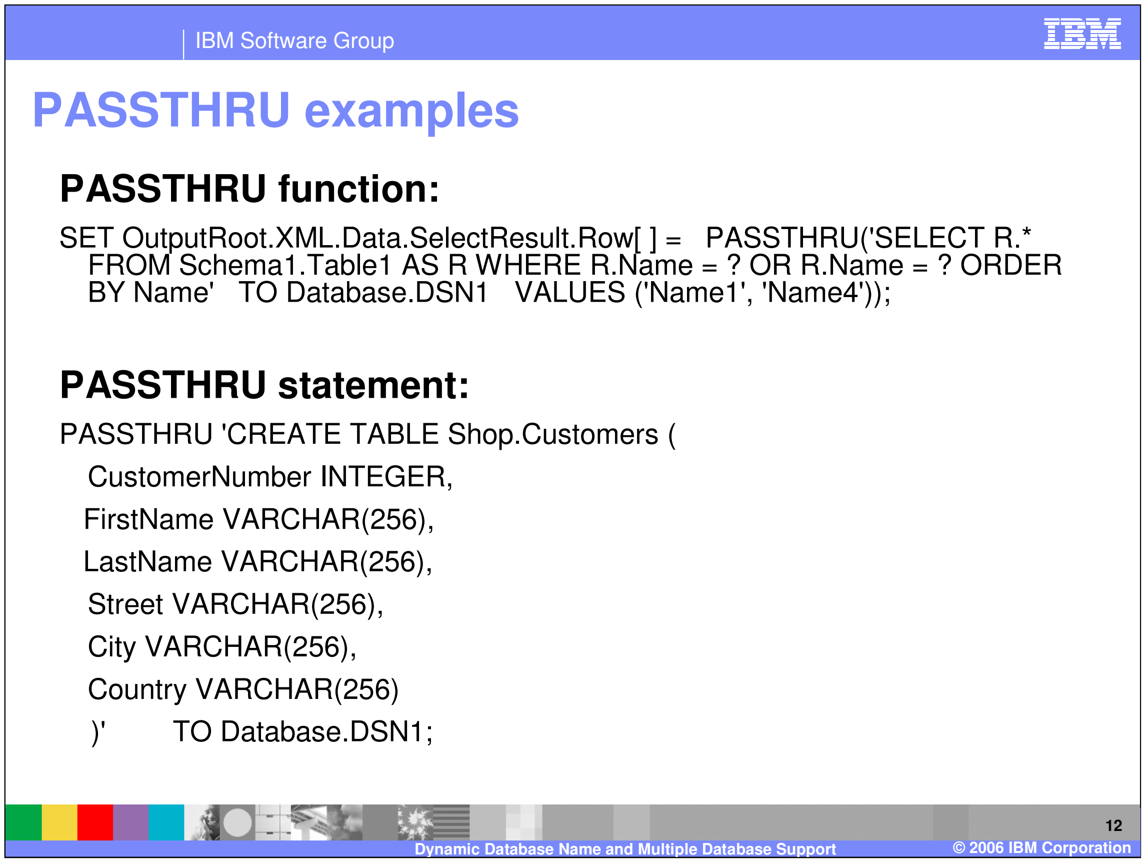
Here is a summary of the Table Reference Syntax provided in V6. The additional elements are the expression elements under DataSourceClause, SchemaClause, and TableClause.



Syntax

- Existing Insert, Update, Delete, and Select syntax basically unchanged
  - Now allows the additional expression statements with the additions to the Table Reference syntax
- Call statement syntax now allows database identification
  - Identifies which database contains the stored procedure to call since more than one database can be used
- Passthru statement and Passthru function allow database reference

The syntax for insert, update, delete, and select statements is basically the same as in previous releases, except for the addition of the expression elements in the Table Reference syntax. The “Call” statement now additionally allows the reference to data source name and schema name expressions. The Passthru statement and the Passthru function both allow a database reference.



PASSTHRU examples

PASSTHRU function:

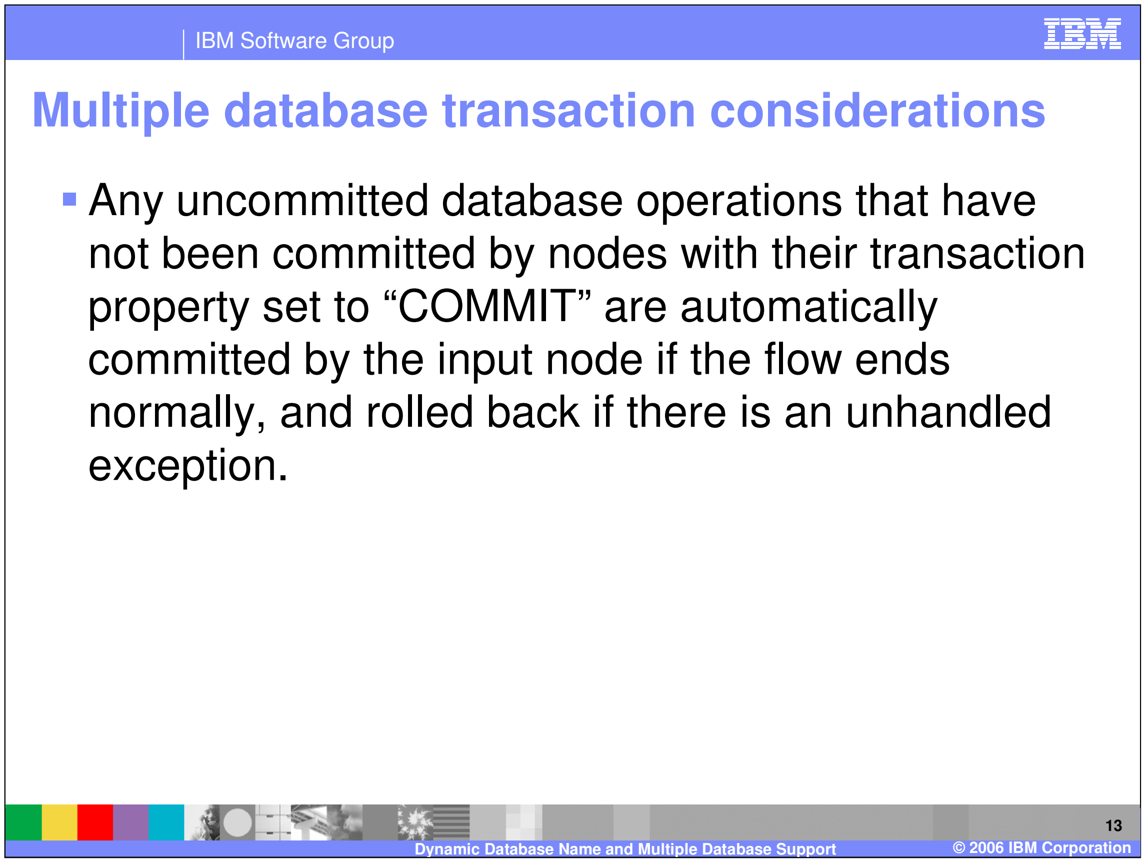
```
SET OutputRoot.XML.Data.SelectResult.Row[ ] =   PASSTHRU('SELECT R.*
   FROM Schema1.Table1 AS R WHERE R.Name = ? OR R.Name = ? ORDER
   BY Name'   TO Database.DSN1   VALUES ('Name1', 'Name4'));
```

PASSTHRU statement:

```
PASSTHRU 'CREATE TABLE Shop.Customers (
   CustomerNumber INTEGER,
  FirstName VARCHAR(256),
  LastName VARCHAR(256),
   Street VARCHAR(256),
   City VARCHAR(256),
   Country VARCHAR(256)
   )'       TO Database.DSN1;
```

The PASSTHRU function evaluates an expression and executes the resulting character string as a database statement, returning a result set. This example performs a SELECT on table "Table1" in schema "Schema1" in database DSN1, passing two parameters to the WHERE clause and asking for the result set to be ordered in ascending name order. The result set is assigned to the SelectResult folder.

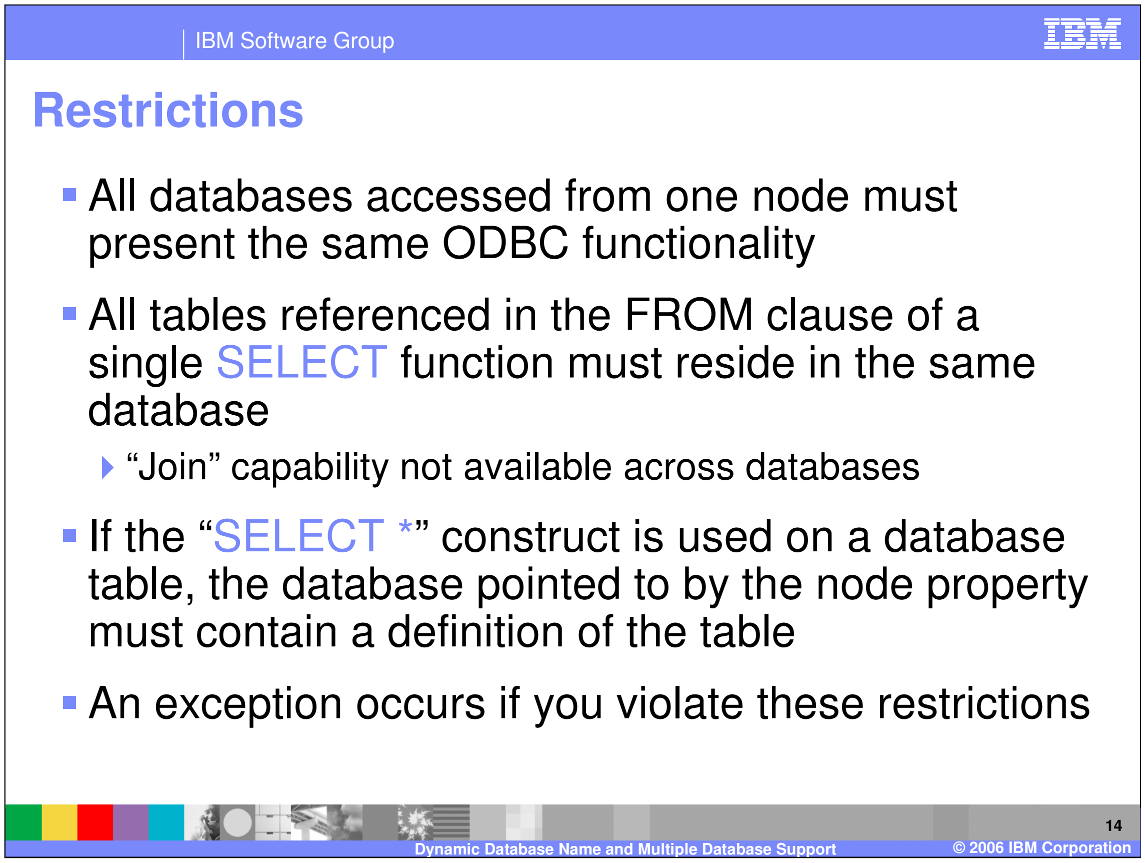
The PASSTHRU statement evaluates an expression and executes the resulting character string as a database statement. This example creates the "Customers" table in schema "Shop" in database DSN1.



Multiple database transaction considerations

- Any uncommitted database operations that have not been committed by nodes with their transaction property set to “COMMIT” are automatically committed by the input node if the flow ends normally, and rolled back if there is an unhandled exception.

Please note these considerations regarding the transactionality of database operations. Any database operations that have not been committed by nodes with their transaction property set to “COMMIT” are automatically committed by the input node if the flow ends normally. Likewise, under the same environmental circumstances, the database operations are rolled back if there is an unhandled exception.



Restrictions

- All databases accessed from one node must present the same ODBC functionality
- All tables referenced in the FROM clause of a single SELECT function must reside in the same database
  - "Join" capability not available across databases
- If the "SELECT *" construct is used on a database table, the database pointed to by the node property must contain a definition of the table
- An exception occurs if you violate these restrictions

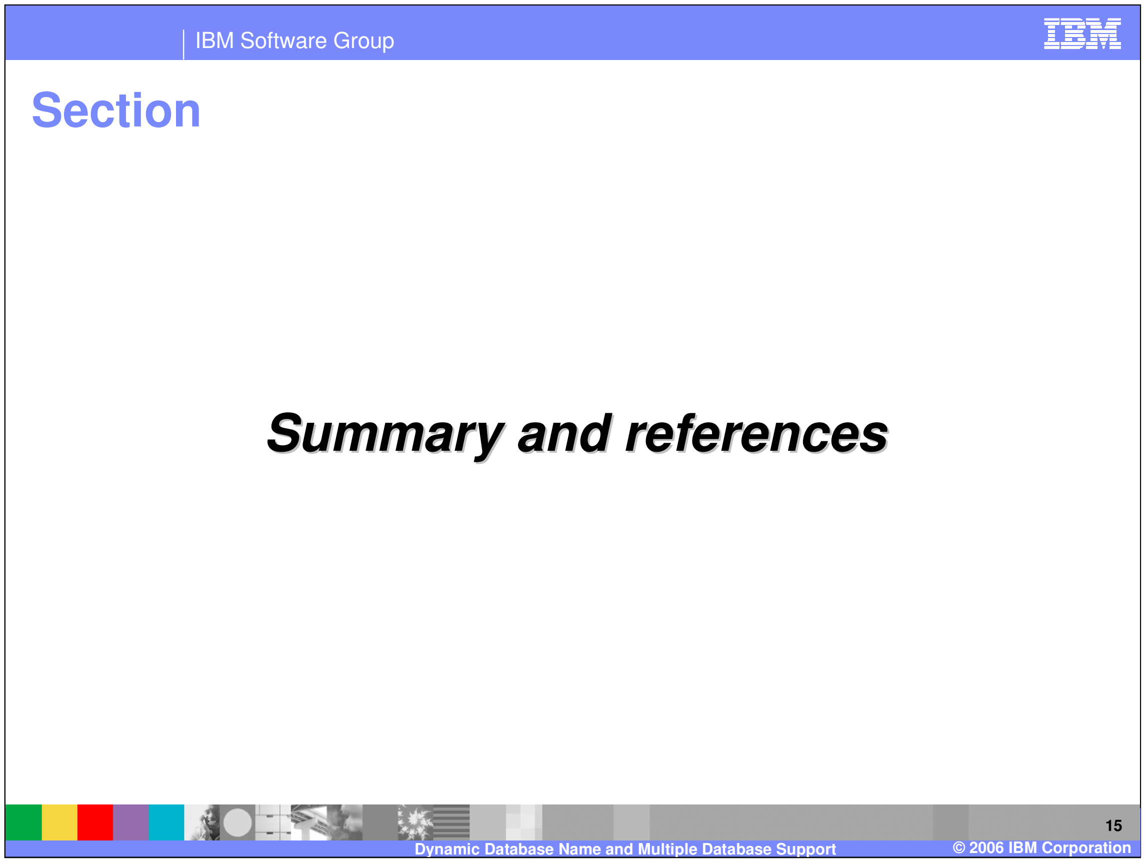
There are restrictions you should consider when using multiple database access within a single ESQL node.

All databases accessed from one node must present the same ODBC functionality.

All tables referenced in the "FROM" clause of a single SELECT function must reside in the same database. In other words, a logical "JOIN" capability does not exist across databases in V6.

If the "Select *" construct is used on a database table, the database pointed to by the node property must contain a definition of the table.

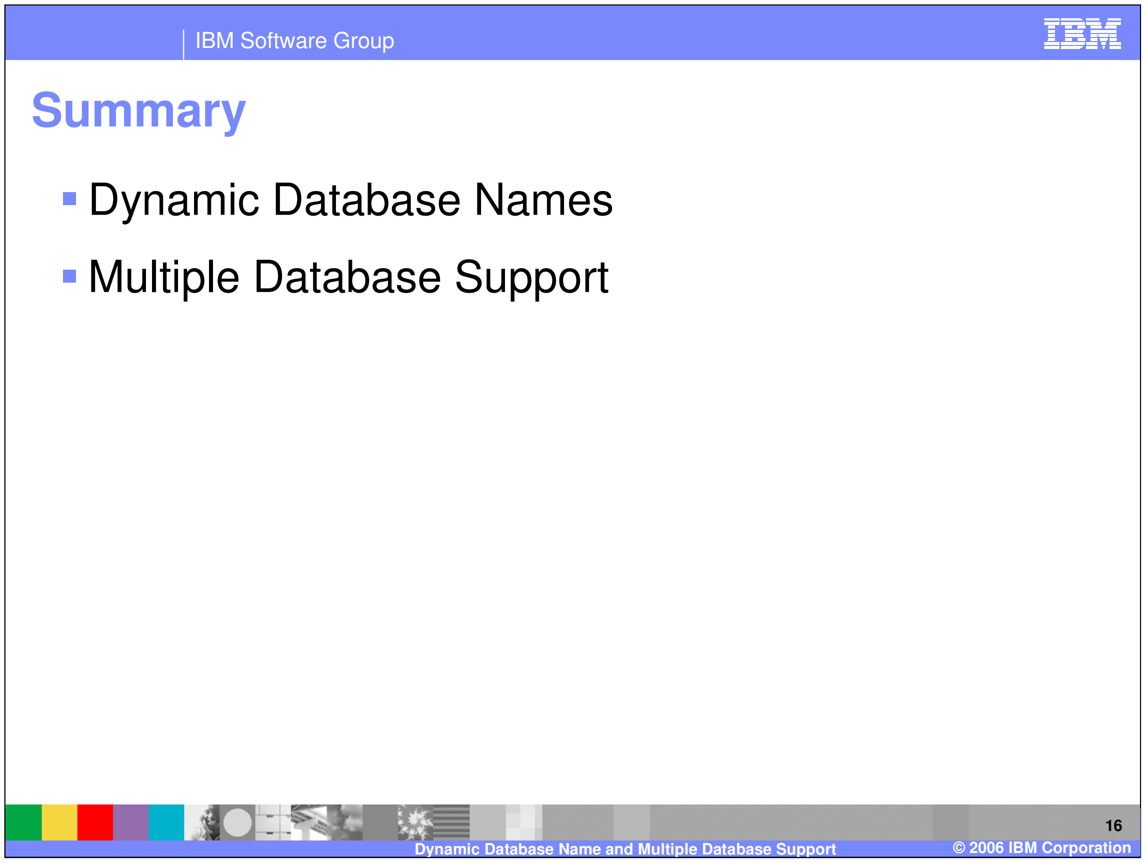
You will receive an exception if you violate these restrictions.



Section

Summary and references

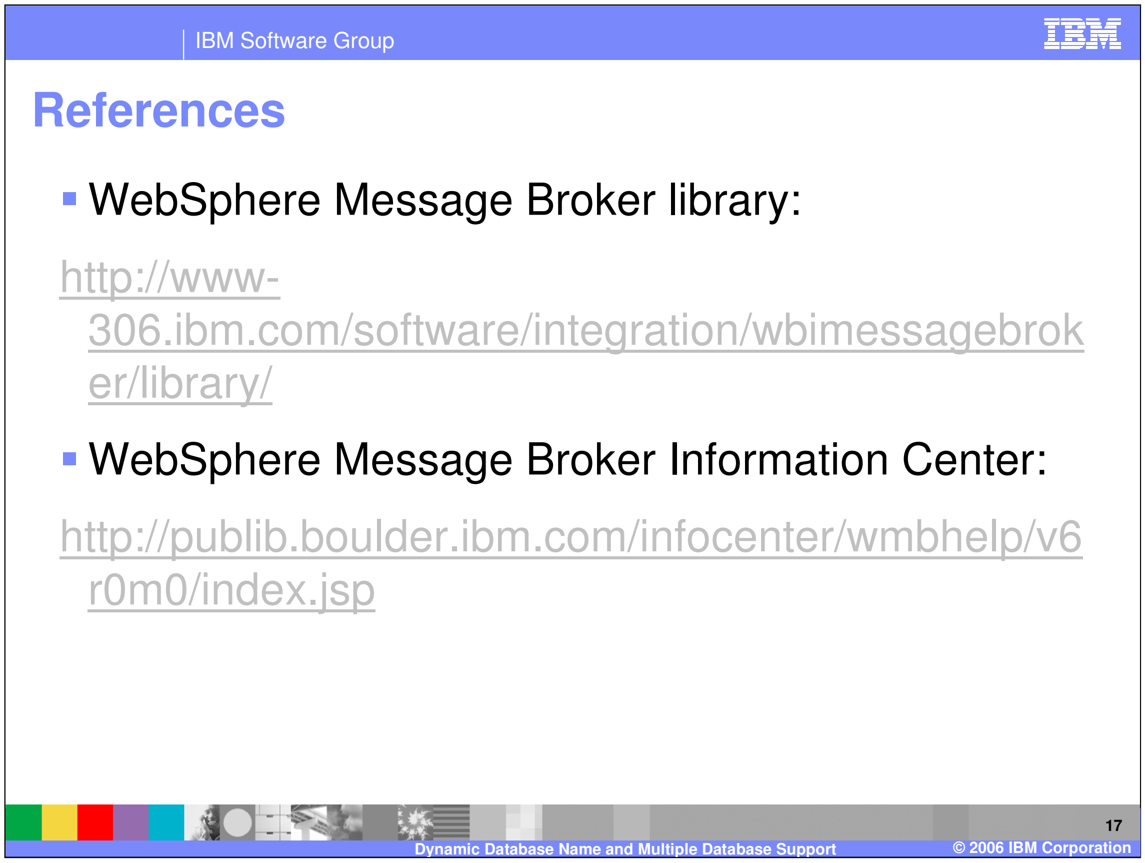
The last portion of the presentation contains a summary and references.



Summary

- Dynamic Database Names
- Multiple Database Support

This presentation discussed Dynamic Database Names, a new feature in Version 6 that enables you to use expressions that can be resolved to the data source name, schema name, and table name, which removes the requirement to hard-code this information in your ESQL statements. This presentation also discussed the new Multiple Database Support feature in Version 6, allowing you to access more than one database in a single ESQL node.



References

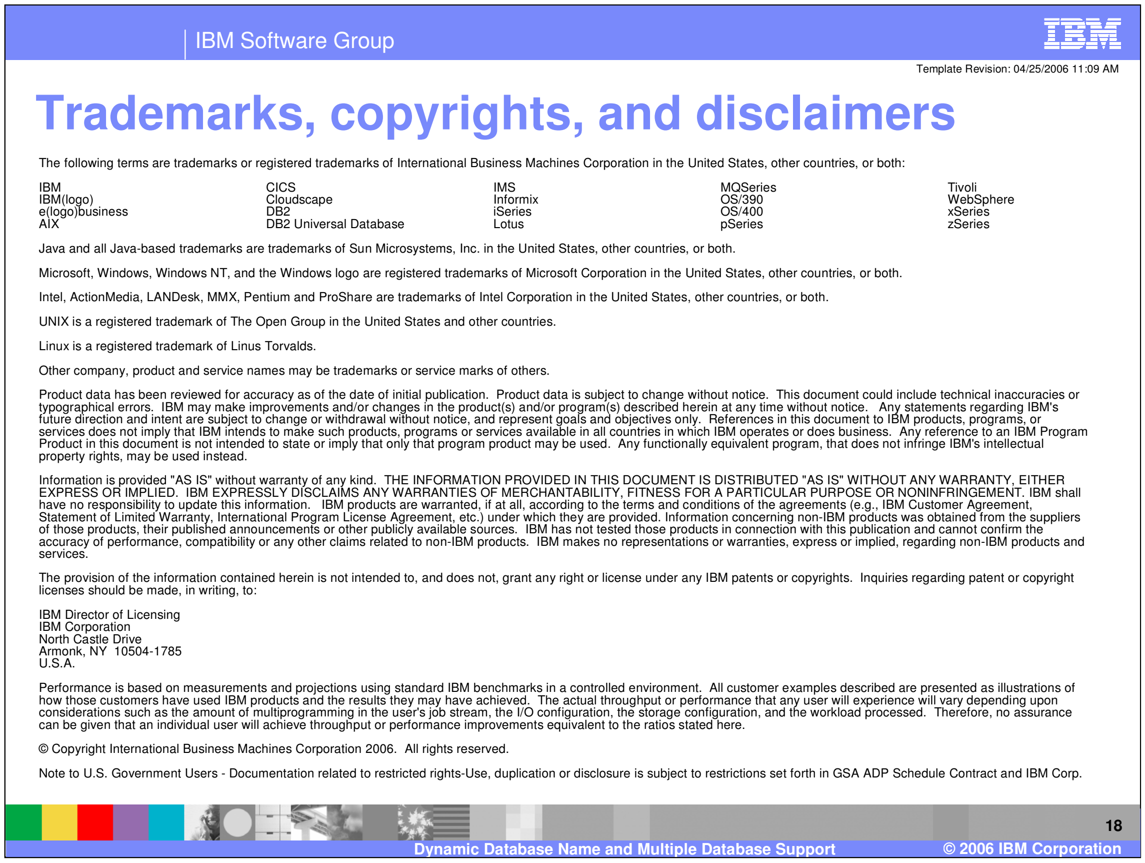
- WebSphere Message Broker library:

http://www-306.ibm.com/software/integration/wbimessagebroker/library/

- WebSphere Message Broker Information Center:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp

References



Template Revision: 04/25/2006 11:09 AM

Trademarks, copyrights, and disclaimers