

-continued

<u>Attributes</u>		
Tag	Card	Comment
AutoCommands	Opt	When set to true, allows the mapping framework to autogenerate SQL for Command instances that are absent. In the case of Views, it is assumed that the user has implemented the view with triggers if necessary appropriately on the server. When set to false, absent commands are treated as "disabled". Default is "false."

[0216] Sub-Elements

[0217] There are no sub-elements for this type.

[0218] BasedOn Semantics

[0219] The BasedOn structure provides a shortcut so that the CustomTable can inherit its Column definitions from an existing structure rather than listing them explicitly. The BasedOn structure is used to automatically generate any Command instances that are not explicitly overridden by the user. A CustomTable that is BasedOn inherits the Relationships that are defined in the structure on which it is based. These Relationships can be referred to by name in a RelationshipMap as if the CustomTable were the base relational structure.

[0220] BasedOn also gives the user control over the semantics of missing Command instances. When AutoCommands is "false" specific Command instances that are not explicitly overridden are "disabled".

[0221] 4. Columns

[0222] There are two types of Columns: Columns that are inherited via BasedOn and Columns that are explicitly declared. BasedOn Columns are identical to the columns that are specified for a physical table or view, however the user will never actually see the syntax for these columns in the context of the Custom Table. The relevant point is that when AutoCommands is equal to true, the SQL that is generated is identical to the SQL that would be generated against the Base Table so the semantics of the additional annotations (such as default values, AutoIncrement, ReadOnly, and AllowDBNull must be preserved).

[0223] Explicit Columns are explicitly declared columns in the CustomTable, and are abstractions for a Command procedure parameters and/or result columns. Because these are not physical columns on a table, they do not allow additional annotations such as AutoIncrement, Read-Only, etc. They only allow a name and type. In cases where parameters to a routine are declared explicitly (Stored Procedures, UDFs) the type of the column is convertible to the type of the parameter. In the Inline case (no declarative parameters) the type of the column is assumed to be the type of the parameter.

[0224] If a BasedOn is also specified these Columns are appended to the BasedOn columns as "Extended Columns". Extended columns can simplify scenarios that fall into the "Single Complex Mapping" scenario where one (or more)

commands may have more columns than the BasedOn structure. Any auto-generated commands use only the BasedOn columns, while Referenced or Inline commands use BasedOn columns plus the ExtendedColumns. It is up to the Command author to perform the appropriate bindings (if default binding is not sufficient).

[0225] 5. Condition Element

[0226] The Condition element is a shorthand notation for a simple Inline QueryCommand with the additional property that the value used in the predicate is expressed declaratively in the syntax and can be exposed to the Target domain via the mapping interface. The scenario in particular that requires this functionality (aside from being a nice shorthand for the user) is an inheritance scenario where multiple types in an inheritance hierarchy map to the same physical table, and on read they want to map to a particular view of that table.

<u>Attributes</u>		
Tag	Card	Comment
Column	Req	This must be a column in the scope of the CustomTable on which this condition is applied
Value	Req	The literal value to add to the predicate

[0227] Additional operators can be supported by adding an "Operator" attribute to the Condition with the "=" operator as the default for backwards compatibility. In addition, if multiple Conditions are specified they are combined using the AND operator.

[0228] 6. Commands (Query, Insert, Update, Delete)

[0229] Commands are where the user can customize which relational structure or inline SQL is used to perform each of the CRUD operations. The structure of Commands assumes that the most common CustomTable scenarios will override CUD Command instance by referencing database procedures and/or functions (rather than inline SQL), "inlining" is a specialized behavior. InlineCommands are an optional child element of the other Command types to reduce confusion in the common case by separating meta-data that only applies to the Inline scenario.

[0230] Query, Insert, and Update Commands can return Result Sets and Output Parameters. In order for OutputParameters to be surfaced in the target domain they must be bound to a custom table column that participates in a FieldMap. In the QueryCase, the value of the output parameter is copied to every row in the result set. Since Insert and Update only return at most one row in the Result Set the value of the output parameter is treated just like a column value.

[0231] Because Output Parameters are treated as though they are part of the row(s) being returned from the Command, a result column cannot bind to the same Custom Table Column as an Out or In/Out Parameter.