Diff Functions

Using Diff Functions to Synchronize Data

The Jitterbit Diff/Synchronize feature can be used in transformations to synchronize data in a single table to any kind of target.

The source must be a database or a CSV file. If a database, the source table must have a set of keys that uniquely identifies a data row. Hierarchical source table relations are not supported.

The diff feature is used to split the source data into three categories:

- added rows
- updated rows
- deleted rows

The first time a diff operation runs, all the source rows are classified as "added". With each subsequent run, only added, changed, or deleted rows will be processed by Jitterbit. This can save significant processing time in cases where a source table has numerous records that change infrequently.

As will seen in the steps below, a "diff operation" usually consists of several chained operations; each successive operation handles adds, updates, and deletes respectively.

To set up a diff operation, follow these steps:

1. Initialize the diff session and define the type of diff algorithm to use. Currently, either "chunked" or "ordered" are supported. *Chunked* will always work, but *ordered* may be faster if the order of the source data rows is guaranteed.
2. Handle added, updated, and deleted rows. The order that they are handled is arbitrary and depends on the order of the operations that implement the synchronization.
3. Finalize the diff session. This flags the diff process as complete; otherwise, a future diff operation can result in no records being processed.

> **WARNING:** Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Step 1: Initialize the Diff Session

Initialize the diff session by calling the `InitializeDiff()` function with a unique "diff ID" followed by either the `DiffKeyList()` function (for a chunked diff) or the `OrderedDiffKeyList()` function (for an ordered diff) to define the primary key(s) of the source table:

Example

```
InitializeDiff("A-unique-diff-ID-for-each-diff-process");
DiffKeyList("pk1", "pk2");
```

Step 2: Handle Added, Updated, and Deleted Rows

Before each transformation, the functions `DiffAdd()`, `DiffUpdate()`, and `DiffDelete()` are called. Any transformation running after those function calls will be passed the added, updated, or deleted rows respectively. The source of the transformations has to be the same database table or CSV file, but the target can be different in all three cases.

Step 3: Finalize the Diff Session

The `DiffComplete()` function should be called when the three cases have been handled successfully. (In the case of errors, the `ResetDiff()` function should be used instead.) This flags the diff process as complete; otherwise, a future diff operation can result in no records being processed.

Additional Diff Functions

- **`DiffNode`:** For hierarchical sources, specifies the node that is to be used as the repeating node to perform the diff on.
- **`OrderedDiffKeyList`:** Specifies the sort order of the source when the "ordered" diff algorithm is used.
- **`ResetDiff`:** Resets or purges the diff session. Used in error handling to reset the diff session on failure.

On This Page

Search in This Topic

Related Topics

- Cloud Studio
- Functions
- Scripts

Last updated: Dec 19, 2019

- **`SetDiffChunkSize`:** Sets a parameter for the "chunked" diff algorithm. A larger value is faster, but uses more memory.

Diff Examples

For examples using diff in Design Studio, see Diff Functions and Capturing Data Changes with Table or File Changes. These may be extrapolated for use in Cloud Studio.

[Back to Top]

DiffAdd

Declaration

```
void DiffAdd()
```

Syntax

```
DiffAdd()
```

Description

Requests the added records as input for the next transformation that is run.

WARNING: Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// Start by processing the added records in this operation:
DiffAdd();
```

[Back to Top]

DiffComplete

Declaration

```
void DiffComplete()
```

Syntax

```
DiffComplete()
```

Description

Flags the diff process as complete. This method is to be called when the diff process completes successfully; otherwise, the diff process will be left in an inconsistent state. In that case, the next time the diff operation runs, no records will be processed.

WARNING: Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// Flag the diff operation as complete
DiffComplete();
```

[Back to Top]

DiffDelete

Declaration

```
void DiffDelete()
```

Syntax

```
DiffDelete()
```

Description

Requests the deleted records as input for the next transformation that is run.

WARNING: Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// Requesting deleted records for the transformation run
DiffDelete();
```

[Back to Top]

DiffKeyList

Declaration

```
void DiffKeyList(string k1[, string k2,... string kN])
```

Syntax

```
DiffKeyList(<k1>[, <k2>,... <kN>])
```

Required Parameters

- **`k1, k2,... kN`:** String keys that identify the columns in the diff source to be used to uniquely identify each record

Description

Sets the list of keys to be used for uniquely identifying a record in the diff source. This method is typically called in conjunction with the `InitializeDiff()` function.

WARNING: Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// Using a chunked diff with the synchronizing primary keys
// of CustomerId and OrderId:
DiffKeyList("CustomerId", "OrderId");
```

[Back to Top]

DiffNode

Declaration

```
void DiffNode(string nodeName)
```

Syntax

```
DiffNode(<nodeName>)
```

Required Parameters

- **`nodeName`:** A string that specifies the repeating node to be used for the diff operation

Description

For hierarchical sources, this specifies the node to be used as the repeating node that the diff is performed on.

> **WARNING:** Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// Run the diff on the XML element node <Listing>
DiffNode("Listing");
```

[Back to Top]

DiffUpdate

Declaration

```
void DiffUpdate()
```

Syntax

```
DiffUpdate()
```

Description

Requests the updated records as input for the next transformation that is run.

> **WARNING:** Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// Requesting updated records for the transformation run
DiffUpdate();
```

[Back to Top]

InitializeDiff

Declaration

```
void InitializeDiff(string diffId)
```

Syntax

```
InitializeDiff(<diffId>)
```

Required Parameters

- **`diffId:`** A string that uniquely identifies the diff operation from other diff operations running on the system. The string must be the same for each run of the operation in order to identify the diff correctly.

Description

Initializes a new diff session.

The string passed as the `diffId` must be **different** from all other diff identifiers used on the system but it has to be the **same** each time the operation runs. If a diff session is already running, a call to this method will fail. To clear an old diff session (such as in the case of a system failure), call the function `ResetDiff()` once.

This method is typically called in the pre-source script of the first operation that implements a diff /synchronization.

> **WARNING:** Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// Synchronize the order database:
InitializeDiff("Order Database Synchronization");

// Using chunked diff with the synchronizing primary keys
// of CustomerId and OrderId:
DiffKeyList("CustomerId", "OrderId");

// With sufficient memory available, increase the chunk size:
SetDiffChunkSize(100000);

// Start by processing the added records in this operation:
DiffAdd();
```

[Back to Top]

OrderedDiffKeyList

Declaration

```
void OrderedDiffKeyList(string k1, bool isAscending1[, string k2, bool
isAscending2,... string kN, bool isAscendingN])
```

Syntax

```
OrderedDiffKeyList(<k1>, <isAscending1>[, <k2>, <isAscending2>,... <kN>,
<isAscendingN>])
```

Required Parameters

- **`k1, k2,... kN:`** Strings names of column keys to be used to uniquely identify a record in a source
- **`isAscending1, isAscending2,... isAscendingN:`** Boolean values for each column indicating if each column is sorted ascending (`true`) or descending (`false`)

- The number of arguments supplied must be two or more and must be an even number.

Description

Sets the list of keys used for uniquely identifying a record in the source and specifies the key's record order as either ascending (`true`) or descending (`false`).

Use this method (instead of the `DiffKeyList()` function) in cases where the source records are guaranteed to be in a specific order.

Diff processing is more efficient if the source fields are ordered the same way each time. In those cases, no chunk size will be used and memory use will not be an issue. This method is typically called in conjunction with the `InitializeDiff()` function.

WARNING: Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

[Back to Top]

ResetDiff

Declaration

```
void ResetDiff(string diffId, int action)
```

Syntax

```
ResetDiff(<diffId>, <action>)
```

Required Parameters

- **`diffId:`** A string that uniquely identifies the diff operation from other diff operations running on the system
- **`action:`** An integer (either `0` or `1`) specifying how the diff session is to be reset

Description

Resets an existing diff session.

The `action` parameter (either `0` or `1`) specifies how the diff session is to be reset:

- **`0` (Reset):** Completely forgets the latest snapshot and starts over from the beginning. This will force the system to treat all entries as "added."
- **`1` (Purge):** Removes any files left over from a previous diff session. This does not reset the latest snapshot; it only clears stale files left from old, failed, or canceled diff sessions.

This method is typically called when something has been changed in an existing diff process or if a diff process has failed and left in an inconsistent state. It should not be called during normal diff processing. If no diff session is present for this `diff_id`, no action is performed.

WARNING: Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// Purge any files left over from a previous diff session,
// but keep the latest snapshot
ResetDiff("Order Database Synchronization", 1);
```

[Back to Top]

SetDiffChunkSize

Declaration

```
void SetDiffChunkSize(int chunkSize)
```

Syntax

```
SetDiffChunkSize(<chunkSize>)
```

Required Parameters

- **`chunkSize:`** Size in bytes for chunks used while diffing

Description

Sets the chunk size (in bytes) used while diffing.

A larger chunk size will make the system use more memory but it will process the diff faster. The default is 50000 bytes; if you have sufficient memory, you can increase this number.

This method is typically called in conjunction with the `InitializeDiff()` function.

WARNING: Diff functions can be used only on a single agent as diff snapshots are not shared. Do not use in an Agent Group with more than one agent.

Examples

```
// With sufficient memory available, increase the chunk size:
SetDiffChunkSize(100000);
```

[Back to Top]