White Paper

embarcadero®

Simplifying Data Change Tracking with InterBase Change Views

By Stephen Ball, EMEA Technical Sales Consultant (RAD)
& Associate Product Manager (InterBase)

Embarcadero Technologies

Table of Contents

Executive Summary

The new InterBase® patent pending feature "Change Views" introduces a new approach to tracking data changes (with field level granularity) in server as well as personal computer/tablet/mobile databases. It is designed for today's mobile centric world. Furthermore, Change Views offer developers a way to break traditional constraints in their thinking about data change tracking and the life span of data in briefcase models. Through its Change Views technology, InterBase provides easy access to a highly scalable change-tracking engine that developers can easily mold to fit their requirements, regardless of whether they need to keep 1 or 100,000 destinations up-to-date.

Change Views help speed up development and enhance the features developers can provide to their end users by eliminating the challenges posed by several of the traditional data management techniques.

This white paper examines the traditional approaches used by developers for tracking changes and the associated difficulties faced. In this context, the paper outlines the value of Change Views and why it represents a brand new paradigm for tracking and moving data changes, with a focus on:

- Reducing data traffic and the cost savings associated with decreasing volumes
- Providing realistic ways to keep even the largest off-line caches up-to-date quickly
- Empowering agile developers to complete their applications rather than planning and developing change tracking techniques in their application code

Current Industry Practice

Why track data changes?

I want to start by sharing a story with you. Recently, following a talk I gave at a conference soon after the release of InterBase XE7 and Change Views, I was approached by a software engineer who developed applications for a pharmacy chain. He highlighted that keeping all the pharmacists updated with the latest drug information in a timely manner was an enormous challenge, but critical to ensure that prescriptions were dispensed safely. Each pharmacy had a local copy of this data as they had to ensure 24x7 operation, even if the network to their central office was down.

The only way to keep each pharmacy updated with the data is to either constantly copy down the entire database or track the changes and send the deltas down after they have happened. This anecdote illustrates the need to track data changes.

TYPICAL APPROACHES TO TRACKING CHANGES

When it comes to tracking changes, developers and change tracking technologies have typically centered around one of two approaches

- Setting a time stamp field at the point data is changed
- The use of log tables to record changes

Each change tracking technique has its strengths and weaknesses, so it's important to understand these to fully appreciate how Change Views is different.

TIME STAMP FIELDS

The theory behind using a time stamp/date time/time field is simple - as the data is inserted or updated a dedicate time stamp field is renewed to mark the current date and time. This provides the applications referencing the data a way to filter the records, based on the time stamp. Searching for timestamps that are greater than the last query time in theory then provide the changed records only.

There are a number of challenges when using time stamp fields to track changes in records:

- Tracking deleted records is not possible via a time stamp leading to deletes not cascading to other copies of the data.
- Updates to time zones (e.g. daylight saving or PC clock errors being corrected) can lead to time stamp conflicts.
- Working over multiple time zones can prove difficult - e.g. an update in New York and London at the same time would contain very different time stamps. A standardised time needs to be used to enable this approach to work, but it will not always be possible.
- Inserts and updates normally require tables to be locked when writing the changes; otherwise the user making the amends could potentially miss revisions made by other users simultaneously.
- Time fields are typically used one per record. To achieve field level tracking individuals need to add additional time fields for every field that is being track, which quickly bloats the database.

In summary, time stamps are really only bullet proof for single user access databases that are single threaded. They are always at risk from clock changes on the machine.

LOG TABLES

Due to the limitations of using a time field to track changes, replication and change tracking engines commonly use logging mechanisms to record the table and record ID when data is inserted, updated or deleted.

The benefit of using logs is that they can record both the 'type' of change (insert, update or delete) and the record changed. Additionally, they offer a 'resemblance' of the sequence in which records are amended. This said, the sequence number is often compromised when concurrent updates are happening, as they need to be stored externally from a database transaction.

Tracking changes to a log table is relatively simple to add to any database that supports table triggers for OnAfter insert, update and delete operations. Thus it has become a well-documented and tried approach.

A byproduct of using this approach is that changed records create new entries in the database. If for example, one million records in a SQL statement are modified, it will cause one million records to change, plus another one million new log records to be created. If then the same one million records are updated so that it's reflected in another field, an additional one million log records are generated. So now there are two million log records for one million changed records. Work off the log in sequence leads to duplicate collection of changes and superfluous network traffic along with unnecessarily expanded databases due to the size of log tables.

Going back to the issue of sequence – it is known that sequence is not guaranteed, so how can one's position be found in the log? A common approach, which also allows purging of the used log data, is to track changes for each destination. Typically this is done by adding a trigger for every destination that only fires based on who is making the change. However this creates another problem for bi-directional data movement. For instance, when databases A and B move changes back and forth. If a change is logged in A, and then made in B, it's logged, picked up and moved back to A – the database that originally logged the change – resulting in infinite circular logging. Fortunately, this is easily resolved by adding logic to triggers that avoid logging when the change comes from a specific user. There may still be occasions where the sequence is out, but at least they are less likely to cause issues and the log can be managed.

Logging each destination has another effect. Referring back to the example where one million records are updated – each additional destination multiplies the log table changes. So by way of example, tracking for five destinations would create five million log records. This slows down the database. Imagine if there are 1000 destinations - a lot of database users and triggers have to be created to avoid sending changes circularly, and a huge number of database admin support would be required to configure, equating to one billion log records to update the same one million entries!

Fundamentally, log tables are not suited to scale in a mobile / Internet of Things (IoT) world to a large number of destinations.

BRIEFCASING

A common term in software development is briefcasing. This is when a centrally managed set of data is stored locally to improve speed and performance of applications, reduce network traffic and associated data movement costs, and allow offline functionality. Briefcasing is often run on subsets of data specific to a user.

The age-old issue with briefcasing is that as soon as the data is collected, it's probably out of date! There has never been a reliable way, without traditional logging, to track what changed in order to update that briefcase or even know if it is out of date.

Briefcasing is a key approach to reduce cost and improve scalability and performance in mobilised solutions.

SUMMARY OF CHALLENGES

Reverting to the pharmacy anecdote above, the pharmacy group the developer supports has around 30,000 distributed copies of the data that is regularly updated. It's understood that time stamps pose a risk that is unacceptable and logging would lead to an unscalable 30,001 updates to the group's centrally managed medicine database every time a record is changed. As this is not scalable from a performance standpoint, it rules out traditional replication approaches. Currently, the only option for the group is to send a two-gigabyte database (around 60 terabytes of data in total) over the network every time an update is processed, leading to other technical challenges and considerable costs. Due to these logistical issues, the group updates the database only once a week, which is not ideal when working with medicine data.

There are additional difficulties when using logging centre around parsing log tables to find changes.

It is highly inefficient in a number of ways:

- Logs often have records appearing multiple times for each destination

- Logs are typically record level and don't provide field level visibility of the change without vastly expanding the log itself
- Where changes are pushed out in log order, individuals end up sending the same data multiple times leading to extra network congestion and costs
- To find the unique list of what has changed requires additional SQL queries, joins and database page reads - not terrible if all the data is stored in memory, but most reads are from disk, which is the slowest part of the database to interact with (i.e. the hardware layer).
- All this takes time and numerous CPU cycles which could be used to improve scale in the system elsewhere
- Logs are typically designed as a single table for ease of management. However allowing multiple users to update the log can lead to governance issues. For instance, should another user be able to find out what someone else is tracking or when specific fields have changed?

CHANGE VIEWS

Here are the top 10 things you need to know about Change Views:

1. CHANGE VIEWS SUBSCRIPTIONS AND SUBSCRIBERS

Consider the following scenario of a newspaper that is distributed by a local shop. The newspaper comes in four sections: sports, business, fashion and celebrity. A reader can select one or more sections of interest and decides to picks sports.

Later in the day, the reader passes the shop, and wants to know the latest scores, so collects the sports section again. However, the scores are only a part of the information provided in the sports section; by getting the entire supplement again the reader is taking extra content that is surplus and unnecessary. This is the briefcase challenge outlined above! How can this situation be redressed?

If each section became a subscription that you could subscribe to, and you could specifically ask for ONLY the changed content since you last picked up your copy of the subscription, then it would be possible to find what has changed in the scores.

Change Views work in this way. A subscription defines what tables and columns will be tracked for changes and what type of change to track (e.g. Inserts/Updates/Deletes).

Once defined, a subscription can be subscribed to by users/applications/devices who have the rights to use the subscription. Each user can subscribe multiple different "devices" to the subscription (e.g. laptop, tablet, phone etc.) to provide individually managed bookmarks in the data, making it easy to keep multiple destinations up-to-date.

2. CHANGE VIEWS CAN SPAN TIME AND CONNECTIONS

Reflect on the newspaper analogy again. If you recollect, the reader actually left the shop after the first visit and returned later. This is an important point for Change Views - the functionality work with a unique pair of database user and device ID to define a subscribers place in a subscription. As the reader re-visited the shop, we were able to pick up the changes just for us based on these two pieces of information.

Having the ability to span time and connections allows Change Views to become a realistic option for working for mobile and Internet of Thing (IoT) scenarios where connections are not always assured.

Some database engines will allow users to find out what has changed to data you have an active query running to, but only during a single database session. If this is applied to the pharmacy example, there would be 30,000 connections needing to be online, 30,000 telephone lines active and every time a query is opened, the data set is accessed again anyway. Change Views eliminates these issues by allowing the updates to be identified even when individuals have worked offline – this is a *must* in, a distributed architecture.

3. CHANGE VIEWS IS HIGHLY SCALABLE

As Change Views can span time and connection, it allows larger datasets to be kept up-to-date (briefcased/mirrored) in remote locations, which helps with the scale of large distributed datasets through reduced network traffic and associated cost. Change Views also benefit in scale by not maintaining log tables.

In the Pharmacy PoC example, log tables would struggle with a single update causing 30,000 log records to be created; with Change Views only the single live record is modified.

Because no log records are required, Change Views has near zero impact on database speed and performance regardless of whether it is used by one or 30,000 target destinations.

Rather than using log tables, Change Views uses a new patent pending approach that expands the ACID compliant technology (that is tried and tested for over 20 years) in InterBase to specifically track what has changed.

Change Views also avoids the issue of circular change notifications as it knows which “device” is making the change (this is passed in at the point a subscription starts), eliminating one of the other key challenges faced in scaling change tracking.

4. Change Views consolidate changes

In the context of the sports section in the newspaper above, imagine the reader is following a hockey match between teams A and B.

- The match starts; the score is A 0 - B 0.
- During the game, A scores a goal and the score is now A 1- B 0. – The score field is updated.
- As the game progresses, A scores another goal, so the score becomes A 2 - B 0. -The score field is updated again.
- While there have been two updates in this sequence, only one field has changed.

Because Change Views tracks on the record, rather than in a log users don’t suffer the issues faced with logging where the same record is logged multiple times leading to the identical data being sent numerous times when parsing the log for updates. This reduces the amount of data sent and removes impact on the server for consolidating data changes. The process is highly efficient when compared to traditional logging approaches.

Change Views records the changes made using a new patent pending approach that allows different users at varied times to identify what has changed, along with the type of change (insert / update / delete). Change Views can even be limited to only track specific types of change, if desired.

5. Change Views track with field level granularity

Change Views provides field level identification of what has changed, or using the hockey match example above, rather than being told there is an update to the match, the developer is able to identify quickly the score has changed.

From a usability perspective too, knowing what has changed is much more useful and provides the basis for much richer business logic to deal with change-related notifications and conflicts.

To illustrate, in bi-directional data movement scenarios (when data may be updated from multiple end points) what happens when two people make changes to the same record? Knowing WHAT has changed on a record allows the developer to identify change conflicts easily and code appropriate action to resolve. If only the changed records are provided, the developer still needs to identify the amends, which without back versions, may be impossible.

Field level change tracking opens the door to a new way of merging data changes.

6. CHANGE VIEWS WORK SMOOTHLY WITH SQL

Change Views runs using the same SQL statement to select the original data and changed data! Once the subscription is active, the SQL statement fetches the data changes that have not been passed in the subscription to date. As Change Views work using a SQL statement, using traditional WHERE, GROUP and ORDER clause is also possible.

In the newspaper scenario, imagine the reader only wants to find out the scores related to Hockey - using a SQL statement with a WHERE clause will do just this. Example: SELECT * FROM SPORTS WHERE SPORT = "Hockey"

If the same statement is run for each connection to the subscription, the reader will only ever receive updates related to Hockey, ensuring that the sports that are not of interest are not downloaded and updated. This flexibility allows fine grain identification of specific information within the larger data that the reader is allowed to view change tracking on; and reduces the amount of records to fetch, further shrinking cost and network traffic.

Using SQL as the base for Change Views provides developers a agile tool that can easily be adapted to changing project demands.

Change Views also introduces some SQL additions with the ability to query for specific type of changes to fields. As an example, asking for the sports where the "…SCORE is not same."

7. CHANGE VIEWS PROMOTE GOOD CHANGE GOVERNANCE

While it is important to track changes, it is also important to ensure only authorised users receive notification of changes.

Consider a Human Resources application that contains salary information. What happens if an employee queries a traditional change log to find staff that has been logged as "changed"? Would this give them an idea of which individuals were getting pay rises and those not? Also, what about a medical system where a patient record contains a flag advising the individual is HIV positive? Should anyone with access to the log be able to find these changes?

By eliminating the change logs, Change Views still tracks WHAT has changed within a subscriber's view of a subscription, based on rights of usage. This way, specific data can be protected from change notification, providing a unique level of support for good data governance.

8. Change Views are multi-concurrency friendly

Change Views is based on an ACID compliant architecture that is tried and tested. InterBase has a history of supporting concurrent updates to the same table by multiple users so when it comes to Change Views; there is a natural extension of this original architecture.

Change Views run based on the snapshot of the database at the point in time a connection is started. Consequently, it is not reliant on time stamps or log tables. Because of the snapshot isolation level InterBase provides, time stamp issues around locking tables or missing changes due to date revisions are completely irrelevant here. In addition, users will never miss the changes that others are making concurrently, they will be collected next time individuals query their respective subscriptions.

9. Change Views is not impacted by time

Change Views does not use time stamps to log the sequence of changes made to data. Change Views uses an internal transaction identification that is sequential and is therefore independent of time. By eliminating the reliance on time stamps, Change Views does not run into problems regardless of the date or time that is showing on the device with the data. The time zone change issues covered earlier are also irrelevant to Change Views.

The remarkable thing about the sequence identification used by Change Views is that it is the top level identification for any transaction, so it is set at the start of the transaction, and is part of the answer to resolving sequence issues that traditional logging approaches have.

10. Change Views is available on servers and clients

InterBase has a history of providing a small footprint database engine that can be embedded easily and silently into applications. Thanks to this heritage, it is ideally suited to move to iOS and Android as well as the traditional Windows, Mac OS X and Linux platforms, making it truly an enterprise platform wide and IoT solution. By embedding Change Views deeply into applications, developers can rapidly introduce powerful change tracking features on workstation and mobile software to simplify uni-directional and bi-directional data movement.

Change Views can also be part of an existing architecture. The Pharmacy example described above actually was to run alongside an existing MSSQL server installation that the business was based on. Using InterBase in this way added new functionality without the risk of changing the company's entire systems. Likewise, InterBase ToGo can be used in mobile applications to track the client side changes only. Also, because InterBase offers full cross platform support, it allows connections to be made from the mobile devices directly to the central server database where local network or VPN connections are in place, further simplifying the task of getting data to hand held devices.

WHAT CHANGE VIEWS ISN'T

To be clear, Change Views is not an auditing system. Change Views does not keep track of all value changes over time. If developers want to get a view of data at a specific point in time, then InterBase Journaling Archive can offers the ability to restore to that time period, however this is really an admin function. For auditing, then a simplified audit log can be created using triggers as this is far simpler than replication of data changes. Change Views can however be used to track deleted records only if that is all the audit requires.

SUMMARY

As illustrated by the pharmacy and newspaper examples above, Change Views provides a new paradigm for tracking changes made both centrally and locally, with fine level identification of exactly what has changed.

Change Views helps speed up development and enhance the features developers can dream to provide to their end users, removing the hurdles faced with traditional data change tracking.

InterBase Change Views provides these features in a secure and scalable manner that is ideal for fast, cost effective distribution of data across multiple platforms while still facilitating fast offline data, which is essential to mission critical applications. Change Views is ideal for re-thinking the briefcase model for distributed data in business/IoT applications.

ABOUT THE AUTHOR

Stephen has led development teams commercially for over 15 years within the UK and across Europe working with a range of blue chip companies including Hilton, American Express, Fitness First, Virgin Active; Stephen is a Charted IT Professional, a Senior Technical Sales Consultant (RAD) and the Associate Product Manager for InterBase, a product he has used commercially throughout his career. He is regularly speaking across EMEA and can be contacted via twitter or his blog.

twitter: @DelphiABall
blog: http://delphiaball.co.uk

REFERENCES & LINKS

InterBase: http://www.embarcadero.com/products/interbase
Twitter: @InterBase

- What is Change Views
  http://delphiaball.co.uk/2015/02/06/interbase-change-views-part-1
- Creating Change Views
  http://delphiaball.co.uk/2015/02/06/interbase-change-views-part-2-creating-change-view/
- Using Change Views
  http://delphiaball.co.uk/2015/02/07/using-interbase-change-views/

ABOUT EMBARCADERO TECHNOLOGIES