Tech Notes

InterBase® SMP: Safeguarding Your Data from Disaster

Embarcadero Technologies

February 2009

Corporate Headquarters
100 California Street, 12th Floor
San Francisco, California 94111

EMEA Headquarters
York House
18 York Road
Maidenhead, Berkshire
SL6 1SF, United Kingdom

Asia-Pacific Headquarters
L7. 313 La Trobe Street
Melbourne VIC 3000
Australia

INTRODUCTION

If you are already using InterBase®, you know that it's a high-performance, full-featured, low-cost, and low maintenance database. With InterBase SMP 2009 edition, new data protection and security features provide some real operational advantages for customers who simply cannot afford to lose data.

In many database deployments, even under battery back-up and RAID environments, if a database corrupts then everything since the last backup will be lost. InterBase SMP 2009 addresses this vulnerability and offers extra protections features for those who don't even have the battery backup/RAID combination for data safety. This provides:

- Orders of magnitude greater reliability using InterBase's Journaling capability
- Lower impact of, and faster recovery from, Incremental Backup

Software developers can also take advantage of extra functionality:

- Batch updates for markedly faster batch operations
- Character Large Object ('CLOB') support

RELIABILITY

Data protection and reliability is a particular focus for InterBase SMP 2009, with specific features for Journals and Journal Archiving.

JOURNALS

Journals are short-lived files that represent all of the changes required to complete all transactions that occur within their checkpoint timeframe.

When you turn journaling on, you specify the checkpoint in terms of the number of changed pages and/or the amount of time that has passed, and also a file size (length).

```
CREATE JOURNAL [<journal-file-spec>]
[LENGTH <number-of-pages> [PAGES]]
[CHECKPOINT LENGTH <number-of-pages> [PAGES]]
[CHECKPOINT INTERVAL <number-of-seconds> [SECONDS]]
[PAGE SIZE <number-of-bytes> [BYTES]]
[PAGE CACHE <number-of-buffers> [BUFFERS]]
[[NO] TIMESTAMP NAME];
```

Journals work like this:

- Forced writes are turned off to the database file, all writes are now asynchronous
- Forced writes are turned on to the journals, all writes are now synchronous
- As the pages are changed, they are written to the journals
- A separate thread is processing these page updates and applying them to the database in a more efficient manner than normally available (clustering page updates, etc).

- At Checkpoint, all changes are flushed to the database to ensure consistency in the database file
- At LENGTH, the journal files roll over to a new file. The old one is kept and reused for the next rollover (it is renamed and overwritten).
- If the server dies before the journal contents are written to the database file, on startup, a short term recovery is started and the database is bought up to date.

Journals increase performance as long as they are kept on a different disk from the main database because of:

- The change to delayed writes for the database
- The greater level of intelligence available to the database when writing pages back to the disc - it is able to amalgamate consecutive pages perhaps from different transactions and write them sequentially
- All writes to journals are being done sequentially rather than randomly

By themselves, journals will primarily provide a mechanism to increase performance with multiple disks. *You would not use journals (by themselves) if you had only one disk.*

Journals provide a far greater performance improvement in situations where the database has been put into forced write mode because of reliability concerns. Typically this is done in small organizations where there is no battery backup or RAID set ups. Forced writes operate ten times slower than delayed writes, thus making journaling an attractive option while retaining the same level of reliability.

JOURNAL ARCHIVES

Journals add to the reliability of your system when Journal Archiving is turned on. Journal archives are designed for long term recovery, and when activated do two things:

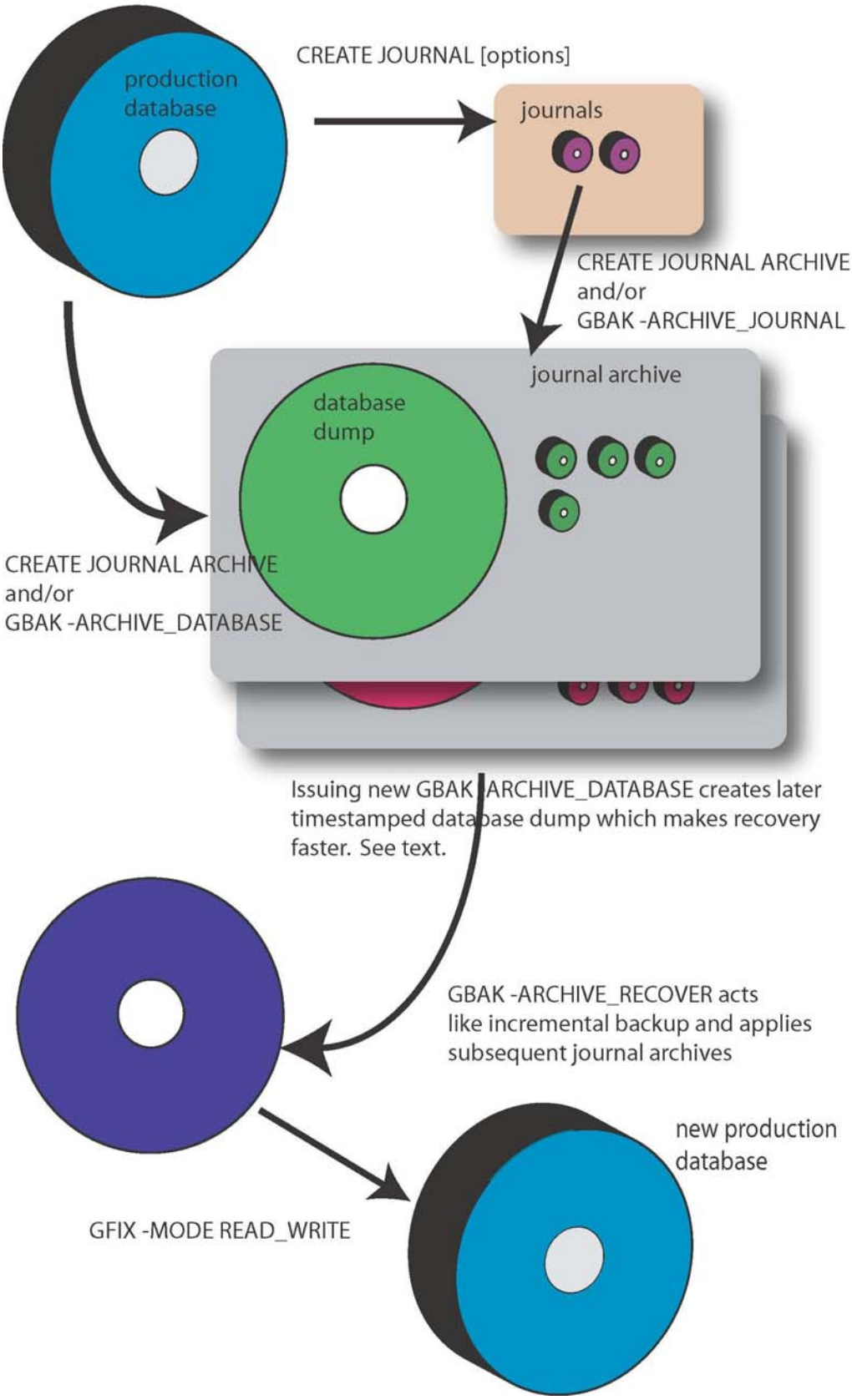
- They take a copy of your database as of the time you created the journal archive (a database dump, the same technology that is used for the incremental backups we discuss later)
- When a journal hits its length, it is archived instead of being "deleted" (re-used). The journal is copied to the Journal Archive directory.

Activating journal archives is as simple as issuing the command:

```
CREATE JOURNAL ARCHIVE [directory]
```

And you are all set to use journal archives.

Journals and Journal Archives



RECOVERING DATABASES

If the database becomes corrupted due to a disc fault or power surge or similar, you can now simply choose to recover the database into a different location. You need to:

- Choose to Archive Recover to a specified location (gbak -archive_recover)
- Set the database to read-write mode (as it will be in read-only mode as above, gfix -mode read_write)

This process will copy the dumped database, apply the journal archives and then apply any journals that are time appropriate. Your database will then be transaction consistent as at the time your database failed.

ARCHIVE RECOVERY IN ACTION

You do not have to wait until the database faults though. The more changes that occur since the time you started the journal archive, the longer it will take to recover (as it has to write all those page changes back to the database). You have two options (or a combination thereof) here:

- Periodic Recovery: You can recover the database constantly if you wish - InterBase will only apply the changes since the last recovery, so you can do this a number of times during a busy day.
- Multiple dumps: InterBase allows you to force another database dump to occur into the journal archives directory. Recovery from a later dump will mean you do not need to apply earlier journal archives.

You can also specify a maximum number of dumps so that InterBase just keeps rolling them over - and you can set an operating system task to take a dump every period of time (2-4 hours for example).

You can also recover to a specific point in time if you wish - you just specify the time when recovery the database and it will apply all transactions up to that point.

ARCHIVE RECOVERY SIDE EFFECTS

The database that you are recovering to is a valid, read-only database, so you can use it for reporting purposes if you are willing to have a lag from the main database (it acts like page based replication in this format) - it will be transactionally consistent due to the way pages are updated to the recovered database during the recovery process the same as they are during normal database writes. You can also use it to issue a GBAK from if you wish.

JOURNAL DEPLOYMENT SCENARIOS

There are a number of typical scenarios that Journaling is deployed in.

SINGLE SERVER, SINGLE-DISC

Why would you do this? Embedded sites are fairly often guilty of running in the following conditions:

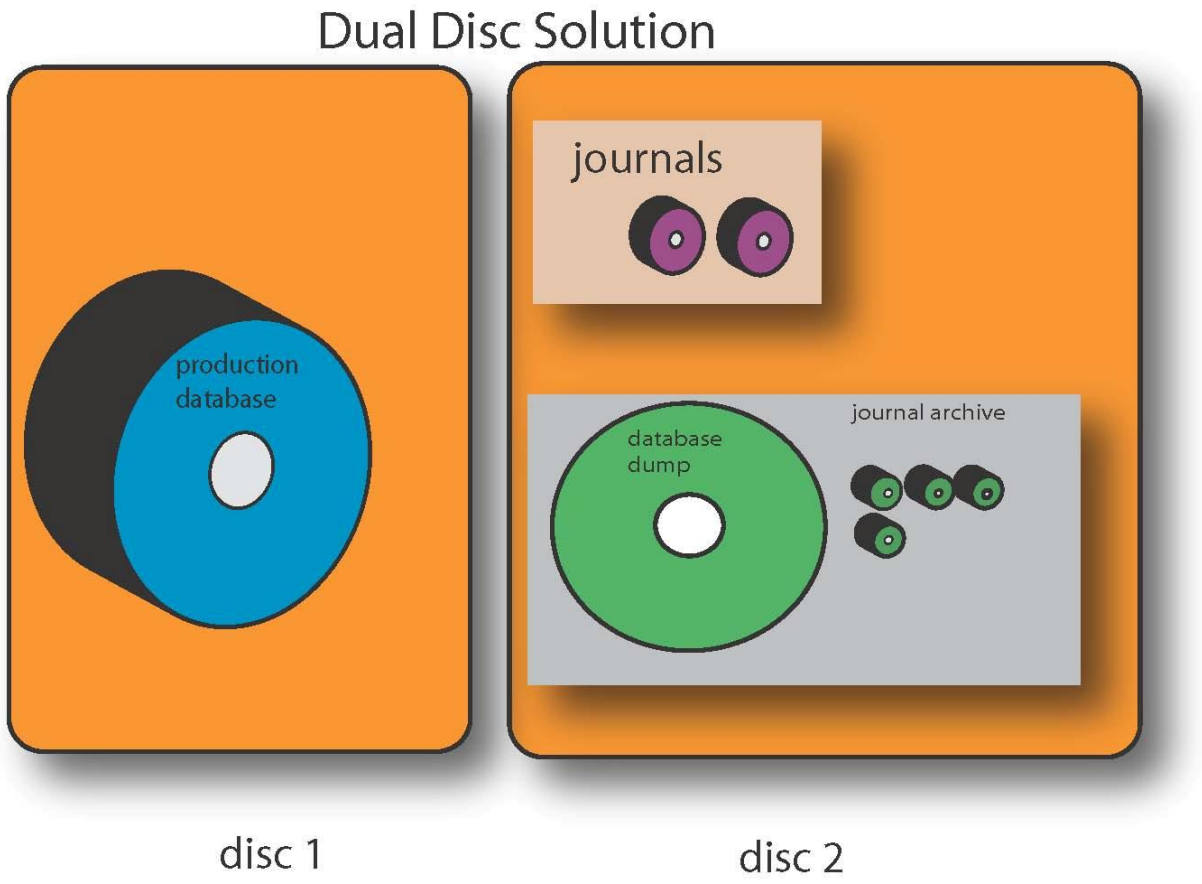
- They do not install RAID of any form
- They have only one disc
- They infrequently do backups

In these circumstances, turning journal archiving on until they do a backup (or you automatically turn the journal archives on and off every day) is worthwhile. It means there will be a duplicate of the database and all transactions that occur. It means that it increases the chances of recovery of the database - and is a process that can be easily automated.

The downsides of this are that the discs' heads are not able to maximize speed for sequential writes into the journals, periodically (depending on checkpoints) taking the heads away to write data to the database image.

SINGLE SERVER, MULTI-DISC

This is a better situation than above. Once disc should be used for the production database, and the other disc can be used to store the journals, and journal archives. Unless both discs fail at the same time, complete recovery will be simple.
This scenario will also allow for the performance increases that are inherent in journals on multi-disc systems.



PRODUCTION SERVER, BACKUP-SERVER

The bigger the system, the more important it is that the server does not go down, and that uptime is as high as possible.

With the archive recovery capability, the typical setup has the journals and journal archives being archived across the network to the backup server. The backup server is then able to run a constant recovery process into a backup database. With this process being automated, the database will typically be ready to receive requests before the network support team has swapped the DNS records so the clients point to the backup server!

JOURNALING SUMMARY

By using the new journaling capability, sites that use InterBase can dramatically increase their reliability - even in single server, single disc situations.

Journaling is typically considered a VLDB capability, but it need not be. Even small servers with the addition of a cheap second disc can achieve dramatically greater reliability for their data where only catastrophic failures (such as destruction of the server) will result in data loss.

INCREMENTAL BACKUP

For Incremental Backup in InterBase SMP 2009, the R&D team decided that InterBase should have as little impact on production as possible and have you up and running again as fast as possible - you then get some goodies thrown in as part of the approach.

INTERBASE SMP 2009'S INCREMENTAL BACKUP

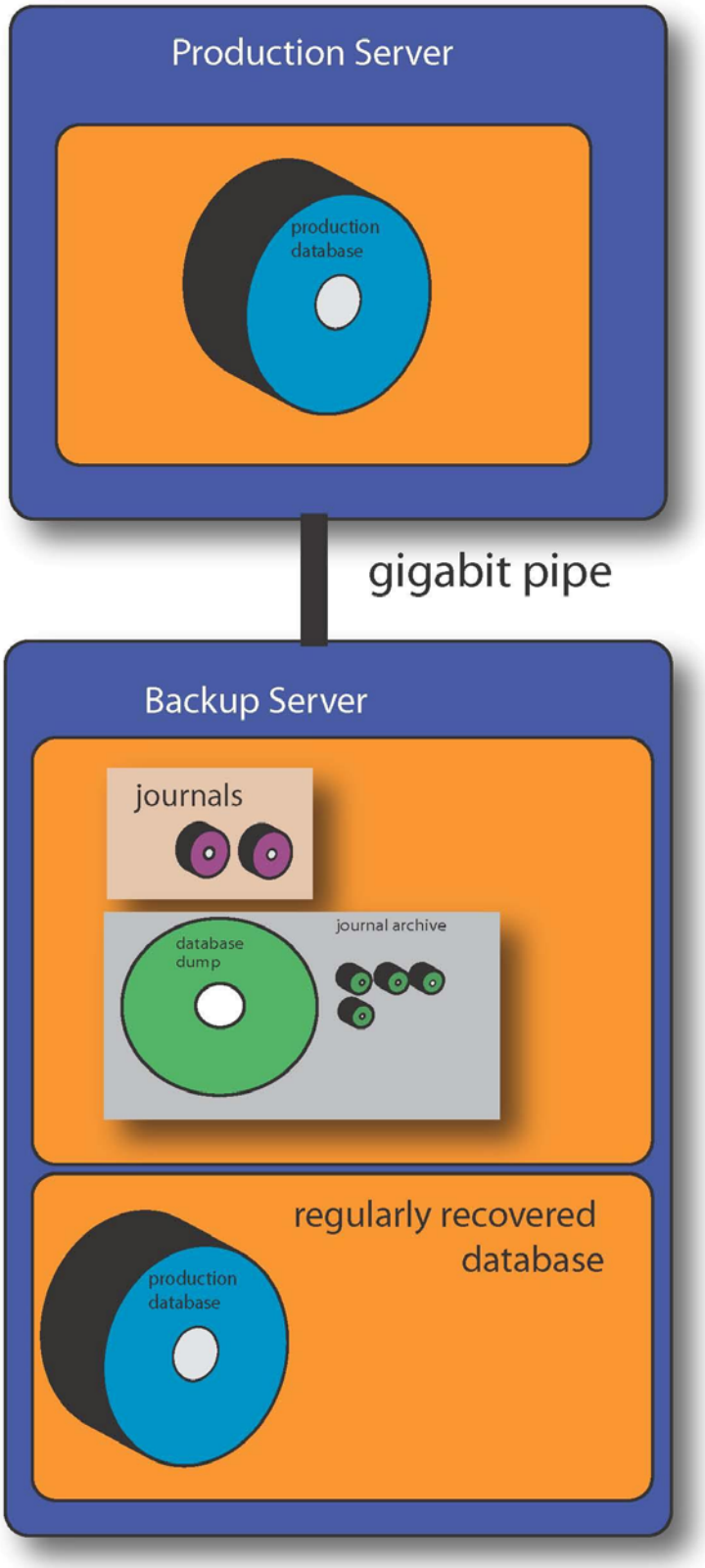
InterBase implements what is probably more accurately described as an "incremental dump" rather than a "backup" in the traditional "GBAK" sense of the word and it falls out of the capability created for Journal Archives.

Once the initial "dump" is made (and you can have as many of these as you like), the incremental backup procedure will simply take the changed pages from the main production database and replicate those changes into the incremental backup database.

Thus, an "Incremental Backup" InterBase SMP 2009 database is ready to go, you can:

- *Turn it to read/write*: by changing its mode, you can make it into a fully operational database in one step (see Addressing Corruption later on).
- *Use it read only*: The incremental backup is a read-only copy of the original database. You can use it for read only queries (such as reporting) if you want to. You can even be doing an incremental backup into it while people are using it as a read only database.
- *Back it up*: using GBAK as you would have the original backup. If you have issued the incremental dump to a different machine, this means the backup process has much less impact on the production server.

Production + Backup Server

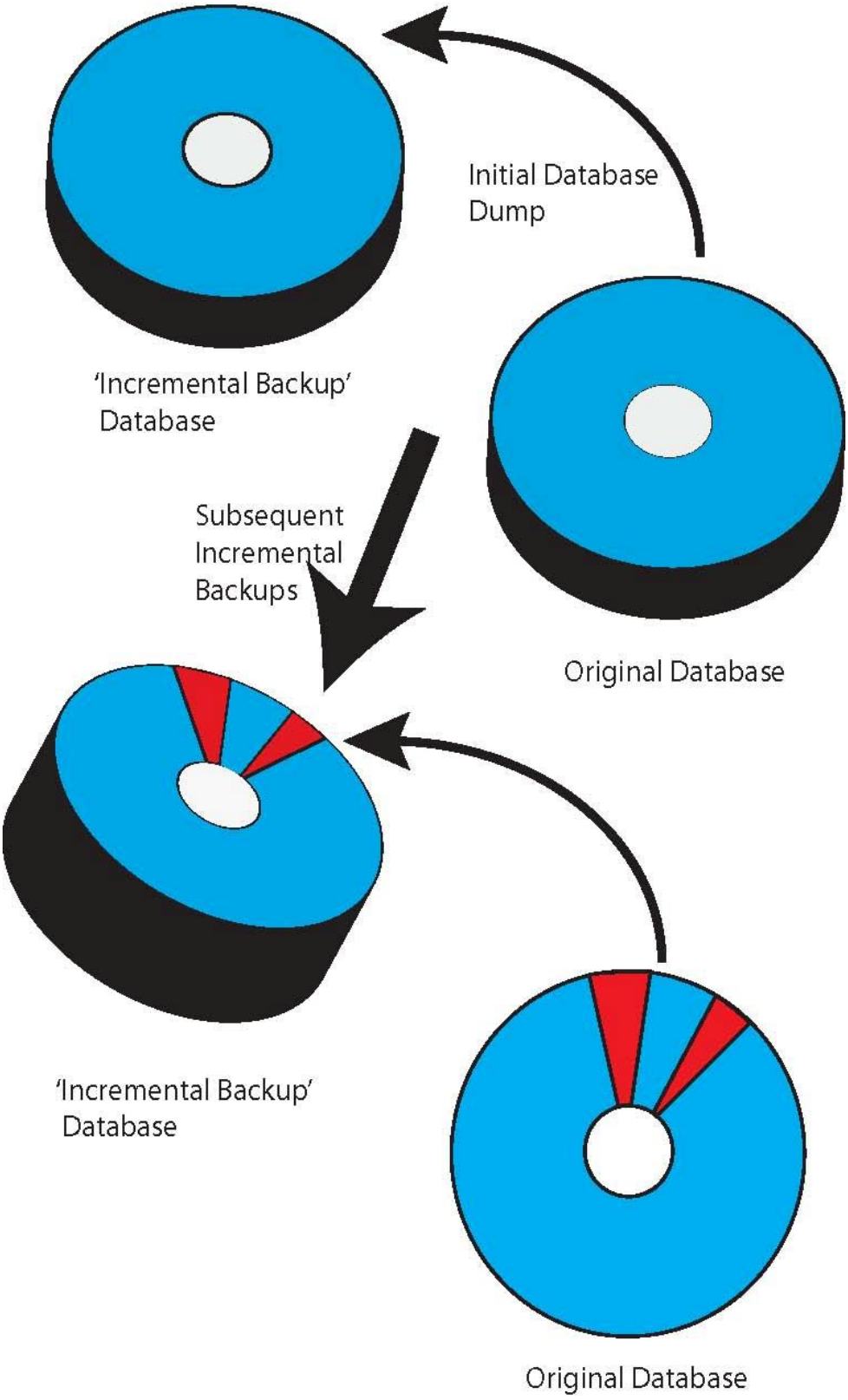


If one (or more) incremental backups are going on, then any pages that change during the incremental backup are stored in a temporary file until they are backed up. The production database file (plus any journals) is always consistent and always the sole source of the current state of the database, and the incremental backup is always transaction consistent as at the time of the backup.

This ensures that even if the incremental backup process is being done by automated scripts, in the event of a system crash, the database will be intact and ready to go immediately; it also means that there is never any period when the database is locked from being used.

So how does InterBase determine what needs to be backed up? InterBase SMP 2009 uses timestamps - so when a page changes, it is timestamped. This was required for "point in time" recovery for journaling.

InterBase 2007 Incremental Backups



ADDRESSING CORRUPTION

So what happens in a failure? How do you get your incremental backup up and operational?

With InterBase you issue a:

```
gfix database.gdmp -mode read_write
```

and you are back in business. If you have journals on this database, InterBase will automatically go and recover them and your database will be as it was when it corrupted.

INCREMENTAL BACKUP SUMMARY

You have to decide which is more complex and difficult for your customer - a faster backup or a pain free restore. For embedded customers, pain free restores tend to be much more valuable.

LESSER KNOWN FEATURES OF INTERBASE

There are a number of other features that InterBase has that people are often unaware of.

PERFORMANCE MONITOR

The Performance Monitor was a new addition in the 7.x series of InterBase. The database tables were first implemented (7.0) and then a Performance Monitor tool was made available in 7.1.

The Performance Monitor provides monitoring as well as logging, and visualization of both of those. The tool is accessed by IBConsole and works once the database has been opened. Network administrators will need to ensure that their machine can talk to InterBase normally as all monitoring and performance is done via querying special database tables.

Monitoring tells you what your database is doing right now, and is able to show you:

- Attachments: these tell you who is attached to the database and what they are doing. You get IP, Host, number of active transactions, their DML, garbage collection, and activity on pages. You can ping them to see if they are alive, shut them down and cancel their queries.
- Database: this shows you the database header page and its updating figures in real time. From here you can flush the write cache and force a reclaim of memory.
- Memory: shows you the memory being used by the database with a breakdown of what memory is being used by what types of operations (e.g. sorting, triggers, transactions, etc).
- Procedures: Shows you all stored procedures in the database and activity they have had on them.
- Statements: What statements are currently running, memory being used, affect on the garbage collector, and page impacts. You can see the statement itself, cancel it, find out who issued it (attachment), and find its associated transaction.
- Transactions, Tables, Views, Triggers: all relevant information can be found out about these database objects in the Performance Monitor.

QUERY CANCELLATION

Query cancellation provides you with just that facility - cancelling queries. This is particularly important if you let users create ad-hoc queries on the database. Surfacing this capability is easy in the IBX components for Delphi, via the C API, or via the Performance Monitor.

The need for query cancellation may be lower now given the ability of InterBase to create close-in-time replicas of the production database via journal archives - intensive reporting could easily be done off a replica instead of the main database.

TEMPORARY TABLES

Another InterBase only feature (since 7.5), global temporary tables release the developer from having to create permanent base tables and ensuring that they are cleaned up (even when the user may abort the operation or connection to the database may be lost).

InterBase global temporary tables allow you to specify at what point to delete the data inserted into the temporary table - at transaction commit or at disconnection. This can be changed during the course of the session if necessary, and it can vary from table to table. The syntax is simple:

```
CREATE GLOBAL TEM¬PORARY table (<col_def> [, <col_def> |
<tcon¬straint> ...])
[ON COMMIT {PRESERVE | DELETE} ROWS];
```

META-DATA SECURITY

Meta Data security was added in InterBase 6.5 - it essentially allows you to control access to your meta-data. This would prevent people from updating or changing your meta-data (table structures and so forth) right up to preventing any ability to actually read the structure of the meta-data, thus preventing people knowing what the structure of your database was (and thus preventing intellectual property theft in terms of your database design). However, this also prevents ad-hoc querying.

PER USER DATABASE SECURITY

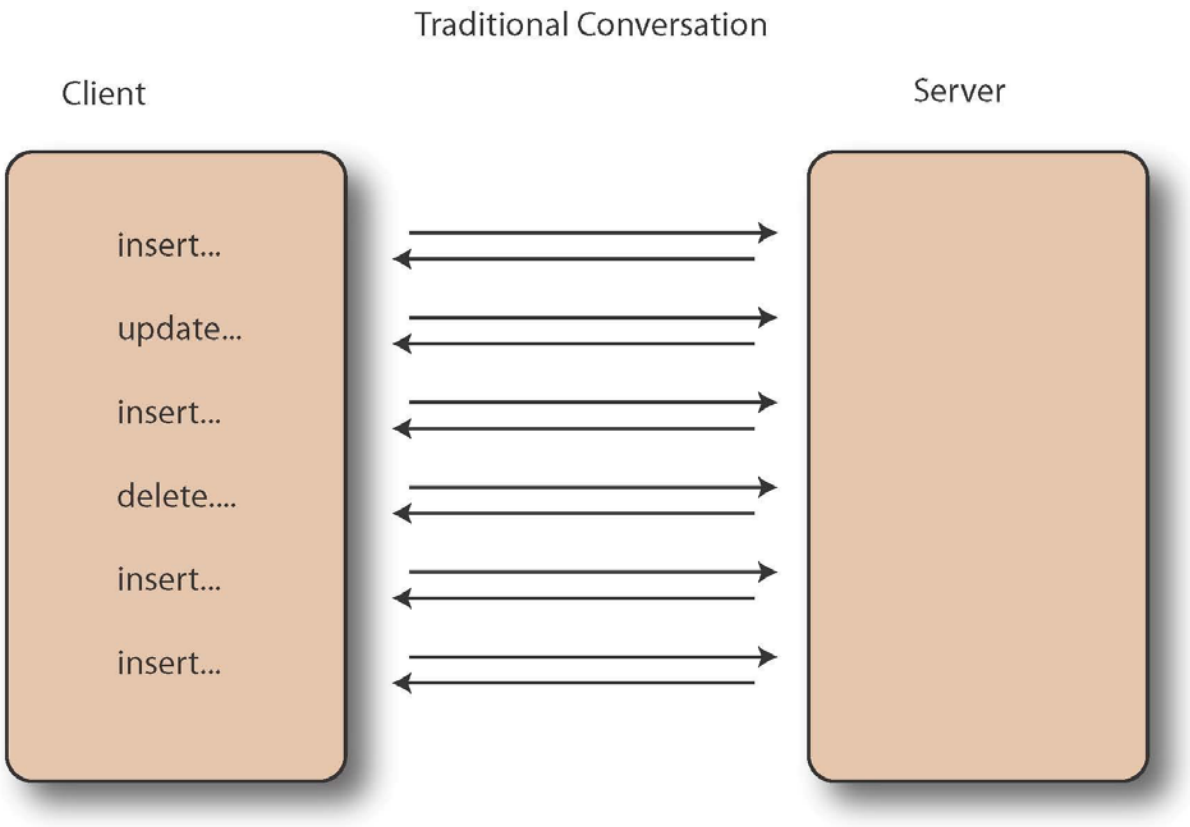
When a database is created, you may decide that instead of having the list of users who can access the database inside the central database infrastructure, they can be stored in the database instead. This is useful when you use InterBase to store users and manage permissions to meta-data through your own applications - especially in ad-hoc reporting situations.

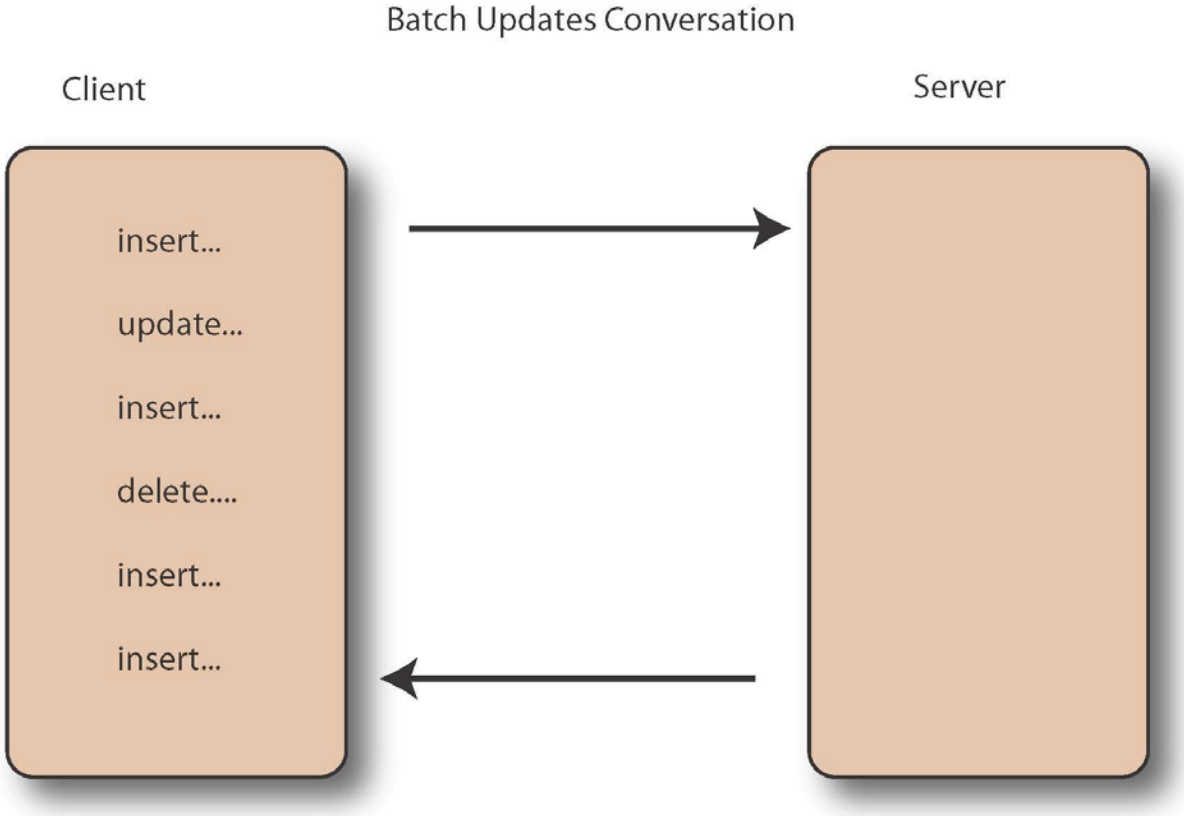
It also means that as the database is backed up and restored (including in a failure situation), all the user management comes along with it, and you do not have to set up the central InterBase admin.ib with those users.

BATCH UPDATES

The Batch Updates feature simply seeks to allow you to put a large amount of DML together in one bundle (up to 2GB) and send it to the InterBase server. The client then gets back an array of responses.

The reason this is faster is that there is no back and forth for each statement between the client and the server, and the server is able to "look-ahead" to make better judgments about what the client wishes it to do.


Traditional Conversation
Client
Server
insert...
update...
insert...
delete....
insert...
insert...


Batch Updates Conversation
Client
Server
insert...
update...
insert...
delete....
insert...
insert...

Support for Batch Updates in InterBase depends on the client libraries used to access it. The IBX components and Java InterClient libraries do support this, as does ISQL, whereas the ADO.NET drivers do not at time of writing.

In ISQL, the usage is:

```
BATCH START;

One of more DML (insert, update, delete) and/or DDL statements

BATCH EXECUTE;
```

A NOTE ABOUT DISC CACHING

One important aspect of database deployment is that typically people will have disc caching turned on for their hard drives. The operating system usually turns it on by default, and the installation of a database application does not turn it off.

It is important to turn this off for a drive with a database on it as the server cannot ensure that the data is being written to the disc and thus ensure the Durability in the ACID properties of the database. This will cause a performance impact but will reduce the possibility of corruption in your database.

To turn this on in InterBase, you need to issue the command:

```
gfix -write sync
```

If you enable journals, InterBase will automatically turn this off (forced writes off, writing asynchronously) and write synchronously to the journals.

Unfortunately, although InterBase attempts to write synchronously, this may not actually work as many discs have on-board caching that exists outside of the operating system caching. You will also need to turn that off if you want this to operate completely correctly. In windows, if you right click on your drive and choose Properties, then select Hardware and select the main database disc, choose the Properties button again and select Policies, you will then be able to turn write caching off.

Embarcadero Technologies, Inc. is a leading provider of award-winning tools for application developers and database professionals so they can design systems right, build them faster and run them better, regardless of their platform or programming language. Ninety of the Fortune 100 and an active community of more than three million users worldwide rely on Embarcadero products to increase productivity, reduce costs, simplify change management and compliance and accelerate innovation. The company's flagship tools include: Embarcadero® Change Manager™, CodeGear™ RAD Studio, DBArtisan®, Delphi®, ER/Studio®, JBuilder® and Rapid SQL®. Founded in 1993, Embarcadero is headquartered in San Francisco, with offices located around the world. Embarcadero is online at www.embarcadero.com.