

Using DB2 incremental backup

Skill Level: Intermediate

[Blair Adamache \(adamache@ca.ibm.com\)](mailto:adamache@ca.ibm.com)

DB2 Development
IBM

[Miro Flasz \(miro@ca.ibm.com\)](mailto:miro@ca.ibm.com)

DB2 Development
IBM

[Dale McInnis \(dmcinnis@ca.ibm.com\)](mailto:dmcinnis@ca.ibm.com)

DB2 Development
IBM

[Susana Uzcategui \(susana@ca.ibm.com\)](mailto:susana@ca.ibm.com)

DB2 Technical Support
IBM

09 May 2002

Updated 29 Oct 2009

Incremental backup is one of the key high availability features in IBM® DB2® 9.5 for Linux®, UNIX®, and Windows® in a data warehouse environment. This article describes how incremental backup works, when you might want to use it, and strategies for ensuring smooth recoveries. This article has been updated for DB2 9.5.

Introduction

DB2 9.5 has many capabilities that provide for the availability and recoverability of databases, these include:

- On-demand log archive

- Backup from split image
- Dual logging
- Veritas cluster support on Solaris

This article discusses one of the most powerful backup improvements for data warehouses — *incremental backup*. Incremental backup means that only changes are backed up. This type of functionality provides you with additional flexibility in designing a backup strategy for some types of database environments.

Why back up just the changes?

Ever save a document in a word processor, make a small change, and then save the entire document again? Consider why you did this — most likely, it was because you wanted to be sure that your latest changes were saved. When you use a relational database, there's something more valuable than the hardware and software you buy to manage the data — the data itself is the ultimate asset. As long as you protect that data, everything else is a bonus.

DB2 has backup and restore commands native to the DB2 engine. Backups can be taken offline (when no users are connected to the database) or online, as the database is being changed. DB2 backups tend to be scheduled by a DBA to run at some interval, for example, weekly, nightly, or hourly. The concept of "I've just changed a lot of things so now I'm going to save my work," does not apply to DB2 (except for users committing data, but that doesn't make an independent copy). Instead, the changes between each backup image are captured in the DB2 logs. These two pieces (the backup image and the logs representing all work since the backup) are the fundamental building blocks of a DB2 disaster recovery plan. Save the backup images and logs some place other than the machine handling the DB2 transactions, and you can restore your data even if the machine that runs your DB2 gets immersed in water and used as a boat anchor.

Captain's log

The concepts of a backup image and logs were designed for a traditional database model with lots of transactions, such as the inventory system in a warehouse. However, as relational databases complete their takeover of the Information Technology world, DB2 is also being used to store data that infrequently changes. The database for an auto parts warehouse is very active (cars break more than software), but the majority of rows in a property tax database would change infrequently because most homeowners tend to stay in their homes at least several years after they move in. There is a mass of data on paper and microfilm that we need to save, and which would be far more valuable in a relational database, but the costs of backing up the contents of the Library of Congress for the 1% of it that changes every n weeks could negate the benefit of storing it digitally.

In multimedia applications, much of the data is stored as large objects (LOBs), which tend not to be logged. So in these cases, even using the backup-plus-log strategy is flawed. If you do log LOBs, the transactional speed of your database is degraded. For this reason, DB2 introduced *incremental backup* — a way to save only the changes made since the last backup.

Advantages of incremental backup

There are two ways for you to use DB2 to track changes and store them in another place for recovery:

- You can have DB2 write every `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `ALTER`, `DROP`, `GRANT`, and `REVOKE` statement to the logs. Then when you need a restore, you can go to the last database backup and have DB2 run through the log and re-create all the changes (something like Sherlock Holmes reconstructing a crime by following the footsteps of each suspect). This method is most efficient in an environment where a large number of transactions are taking place.
- The second option is to have DB2 save a copy of each page when it changes. This is how incremental backup works.

If your database is highly active, it doesn't make sense to keep a copy of each page when it changes. You may end up having a copy of every page in the database (essentially a new backup image), which defeats the purpose of tracking only incremental page changes. Also in this case, it is probably faster to log SQL.

On the other hand, if all changes are concentrated in a small minority of the pages, or if most pages don't change at all, saving changed pages in an incremental backup image saves you time and storage when creating backup images. If nothing changes on the page, the incremental backup skips it.

An incremental backup is more efficient for databases with a small amount of transactions because you are saving only the pages that have changed since the last backup, not every page in the database. Your backup and restore operations are faster and your backup images are smaller.

Enabling incremental backup

To specify whether incremental backup is enabled for a database, you use the `TRACKMOD` configuration parameter. This parameter specifies whether the database manager will track database modifications so that the backup utility can detect which subsets of the database must be examined by an incremental backup and potentially included in the backup image.

The TRACKMOD configuration parameter can have one of the following two values:

- **NO** — Incremental backup is not permitted. Database page updates are not tracked or recorded in any way. This is the default value.
- **YES** — Incremental backup is permitted. When update tracking is enabled, the change becomes effective when the first successful connection to the database is made. Be aware that before an incremental backup can be taken on a particular table space, a full backup of that table space is necessary (more details on this follow the example below).

The following example shows how you would enable incremental backup for the SAMPLE database:

```
DB2 UPDATE DATABASE CONFIGURATION FOR SAMPLE USING TRACKMOD YES
```

After you set TRACKMOD to YES, you must back up the database before allowing applications to change data. In other words, you must take a full database backup so that you have a baseline against which incremental backups can be taken. Also, if you later create a new table space in the database, you must then take a backup that contains the table space. This can be either a database backup or a table space backup. After the backup, incremental backups are permitted to contain the new table space.

A range of backup data

There are two types of incremental backup:

- **Full incremental backup.** An image of all pages changed since the last complete backup (whether the backup is a full or table space backup image). For example, the following command performs a full incremental backup of the SAMPLE database:

```
DB2 BACKUP DB SAMPLE ONLINE INCREMENTAL USE TSM
```

- **Delta.** All pages changed since the last backup of any kind (another delta, an incremental, or a full backup image). For example, the following command performs a delta backup of the SAMPLE database:

```
DB2 BACKUP DB SAMPLE ONLINE INCREMENTAL DELTA USE TSM
```

This gives you up to four kinds of backup data you can use to recover a damaged

database:

- **A full backup image.** This is the building block of any recovery strategy, without a full backup image, you can't start to restore. If the backup is taken online, you require the logs of all transactions that transpired while the backup was taking place. Restore a full backup, replay the logs of all transactions since the backup, and your recovery is complete.
- **An incremental backup.** This includes all changes since the last full backup. Restore a full backup, restore the incremental backup, replay the logs since the incremental backup, and your recovery is complete.
- **A delta backup.** This includes all changes since the last backup of any kind. If the last backup was a full backup image, it and the delta provide the most complete backup. If a delta is preceded by an incremental backup image, you need the delta, the incremental backup, and the full backup image on which the incremental is based. If a delta is preceded by one or more deltas, you need all deltas until you reach an incremental backup or a full backup image.
- **The logs.** The logs contain all transactions since the last backup you were able to restore.

There is another dimension to all of this — DB2 supports backups at both the level of the entire database, and for specific table spaces (a more granular level of backup and restore that allows you to restrict a backup or restore to critical table spaces). Backing up and restoring incremental and delta images applies to table spaces as well.

Recovery strategies

Let's first consider the recovery strategy for a database that has a full backup made each Sunday and a nightly incremental backup, as shown in [Figure 1](#). (Logs are not shown in the figures, but are necessary after the final restore.)

- If you needed to restore on Monday, you would restore with the Sunday night backup image, and apply all available logs.
- If you needed to restore any day from Tuesday through Sunday, you would restore the previous Sunday's full backup image, restore the incremental backup image from the previous night, and finally apply the logs that followed the restored incremental image.

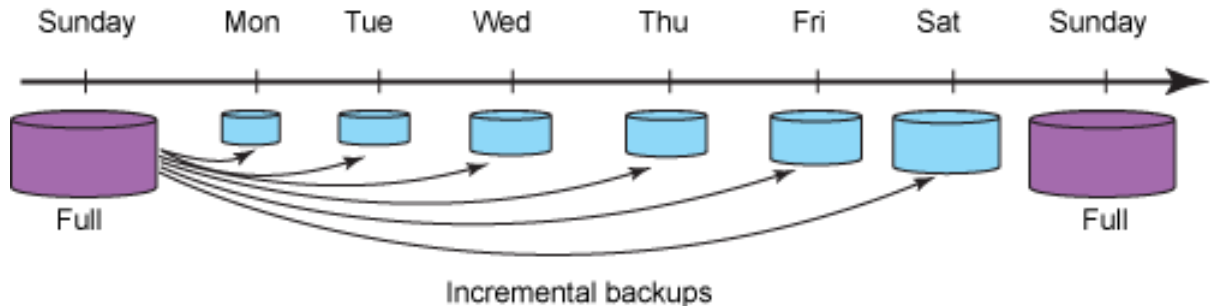
Any recovery needs:

- One full backup image

- Zero or one incremental backups
- One day's logs

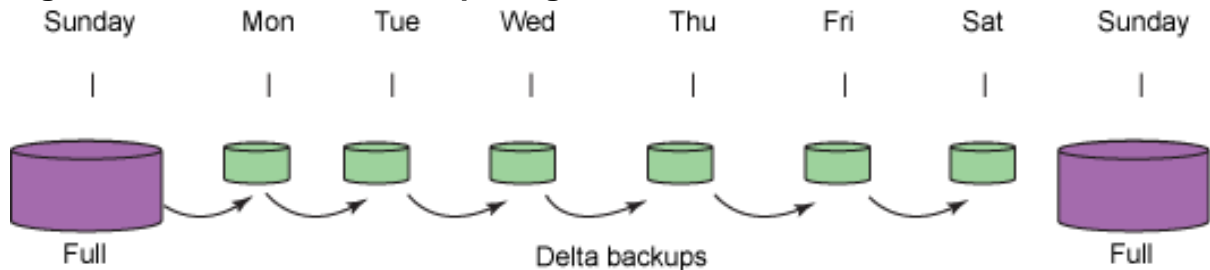
Note that in this scenario each incremental backup gets bigger until the next full backup. This is because as the days progress the incremental backup contains more changed pages. For example, the Saturday backup would include six days' worth of work versus just one day for the Monday backup.

Figure 1. Full and incremental backup images



As shown in [Figure 2](#), the backups would be smaller if you only took a delta backup each night.

Figure 2. Full and delta backup images



By using delta backups, the nightly backups on Monday through Saturday require less storage and finish more quickly. Keep in mind however that when you use nightly deltas, you require all deltas since the last full or incremental backup for recovery. This is different than the strategy shown in [Figure 1](#), which requires you to have only one full backup, one incremental, and logs.

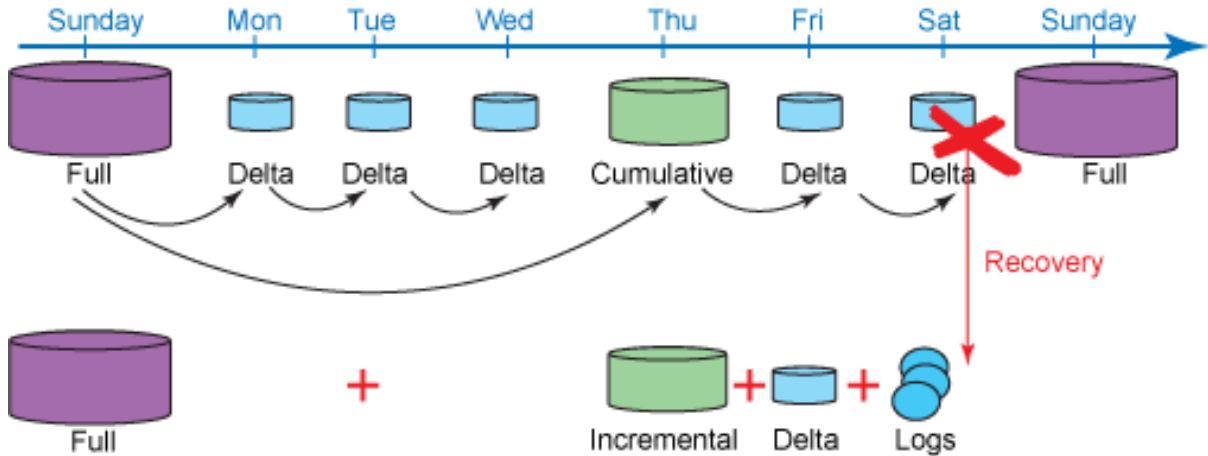
To help you keep track of backups, the DB2 history file keeps track of the backup history for a database. You can use the `LIST HISTORY` command to interrogate the database backup history. The *DB2 Command Reference* (see [Resources](#) section for a link) contains a complete description of the `LIST HISTORY` command syntax. For example, the following command lists all the backup and restore operations made to the `SAMPLE` database:

```
DB2 LIST HISTORY BACKUP ALL FOR SAMPLE
```

DB2 itself uses the history file to determine if any previous images are required for recovery and attempts to restore them automatically. This may lead to a chain of backup images as shown in Figure 2.

Figure 3 shows how recovery requires a full backup image, the last incremental backup, all delta backups since the last incremental backup, and the logs. In the example, once the cumulative incremental backup is made on Thursday, the preceding delta images from Monday through Wednesday are no longer needed. Note that even if one of the incremental or delta images is damaged, as long as you have the full backup image and all the logs, you can still recover in the same manner as you would when recovering without incremental or delta backups. In the example, the Saturday delta image is damaged, so you simply recover to the Friday delta image and use the logs to recover from that point.

Figure 3. Full, incremental and delta backup images



Because DB2 uses the history file to figure out what comes next in incremental recovery, you should use caution when deleting entries from the history file (using the PRUNE HISTORY command). The history file is a piece of metadata that is as important to incremental backup as the system catalogs are to the database.

Automatic restore

If you restore using the INCREMENTAL AUTOMATIC keywords, DB2 determines what to restore. For example:

```
DB2 RESTORE DATABASE SAMPLE INCREMENTAL AUTOMATIC TAKEN AT (SAT)
```

When you specify INCREMENTAL AUTOMATIC, DB2 determines if any previous backup images are required and attempts to restore them automatically. The history file determines the sequence of required backup images. DB2 starts at the last backup image that contains complete copies of all table spaces you are restoring, and then applies subsequent incremental images. Subsequent backup images do

not need to contain all of the table spaces you are restoring.

Before you start a restore, you can also use the `db2ckrst` utility to parse the history file and get a description of the required backup images.

Conclusion

Incremental backup is a good way to protect a database that is largely read-only but has some occasional `INSERT`, `UPDATE`, or `DELETE` activity performed against it. It can also be useful for a database in which changes are isolated to a small subset of the table spaces.

Because an incremental backup strategy potentially relies on four types of data for recovery (full images, incrementals, deltas and logs), keep track of what you're doing. Also ensure that all DBAs supporting the database understand the strategy and know how to read the DB2 history file. If you work in the type of company that can lose archived logs required for disaster recovery, the presence of incremental and delta images gives you more things to misplace.

So whatever your particular situation, you should take the time to establish a backup and recovery strategy. Here are some things to consider when you do:

- Take a full backup before any fundamental change such as a migration, fixpak installation, or a major application change.
- Use the DB2 `db2ckbkp` command to check the integrity of backup images. This command is documented in the *DB2 Command Reference* (see the [Resources](#) section for a link).
- Use the DB2 `db2ckrst` command to query the history file and ensure that your understanding of what you require for an incremental restore is matched by DB2's record.
- Practice a restore so you know how long it takes and where to find all relevant pieces such as backup images and logs. Multiple people need to practice this (after all, you do want to be able to go on vacation or retire some day).
- If you plan to use rollforward point-in-time recovery or to restore table spaces one at a time, be sure you understand the database schema. Restoring a particular table space first does not improve availability if it contains tables with referential constraints to tables in other table spaces.
- If using online backup, recognize that the backup image will be useless without the logs of all transactions that occurred while DB2 was backing up the database.

- Make a decision on what you'll do if a backup ever fails to complete:
 - Will you complete the backup before allowing production to continue?
 - Will you have sufficient logs to bridge back to the previous backup image?
 - Can you adapt your backup strategy if users demand access to the database before a backup completes?

When disaster strikes, DBAs can make things worse if they don't know the strategies and rules you have defined. If they do follow these rules, they can ensure that the disaster is your organization's finest hour.

Resources

Learn

- Learn more in the [DB2 for Linux, UNIX, and Windows 9.5 Information Center](#).
- Access the [DB2 9.5 Command Reference and other DB2 manuals](#)
- In the [DB2 for Linux, UNIX, and Windows area on developerWorks](#), get the resources you need to advance your skills in DB2.
- Browse the [technology bookstore](#) for books on these and other technical topics.

Get products and technologies

- Download [DB2 Express-C 9.7](#), a no-charge version of DB2 Express database server for the community.
- Download a free trial version of [DB2 9.7 for Linux, UNIX, and Windows](#).
- Download [IBM product evaluation versions](#) or [explore the online trials in the IBM SOA Sandbox](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- [Participate in the discussion forum for this content](#).
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the authors

Blair Adamache

Blair Adamache has been working in relational database development and service at IBM Toronto Lab since 1987.

Miro Flaszka

Miro Flaszka has been one of the DB2 Load and Data Management gurus since DB2 Version 5. Prior to V5, Miro worked on extending DB2 functionality for geo-spatial applications as part of his Masters thesis research.

Dale McInnis

Dale McInnis has been part of DB2 Development since Version 1. He owns DB2 Backup, Restore and Recovery, and owns online backup and the generic vendor interface for backup and recovery used by several storage vendors.

Susana Uzcategui

Susana Uzcategui has been working with DB2 for eight years at technical training and support levels. She is certified as DB2 9.5 Associate and DB Administrator.