Read or listen to this article online

Scan the QR code to open this article on XO.PL

Article source:

https://xo.pl/it-made-fun-data-database

DATA & STORAGE

Databases: what Agnieszka does not hide beneath her socks

After the fire at the Central Document Warehouse, the **RAID** rebuild, the **backup** adventures and the **encrypted** safe, it is time to meet the company's most dependable employee: accountant Agnieszka. She remembers invoices by heart, finds old tax records in seconds, and answers “which database engine?” with “Grzesiu knows”. Her office has filing cabinets, a computer, a printer and a drawer full of socks. Naturally, the manager chooses it for his next great revolution. Today we're not learning SQL syntax. We're learning how not to lose, leak or corrupt information — and how to recover it — wherever it happens to be stored.

Chapter one: Agnieszka knows what not to touch

— AGNIESZKA! WHERE IS OUR DATABASE?!
— In the computer, boss.
— WHICH COMPUTER?!
— The grey one. But Grzesiu told me not to go hunting for it myself.
— SO YOU DON'T KNOW WHERE THE DATA IS?!
— I know where the invoices, declarations and accounts are. Grzesiu knows how the software stores them.
— THEN HOW DO YOU MAKE **BACKUPS**?!
— I click the green button and check the message from Grzesiu.
— SO YOU DON'T KNOW WHAT YOU'RE DOING?!
— I do. I'm doing the accounts. Grzesiu handles the databases.

🤣🤣🤣

Agnieszka is a **professional lucky person**: sometimes technically confused, excellent at her profession, and clever enough to call the right person when things get serious. That's a sensible **security** principle: **not everyone must be a database administrator, but someone must be responsible for protecting, backing up and recovering the data**.

A database is an organised collection of information and mechanisms to store and use it. It may run on a dedicated server, in a local file, inside an application, while an ordinary spreadsheet can serve as a simple record system. The tools differ considerably, but invoices and customer records need protection regardless of file extension.

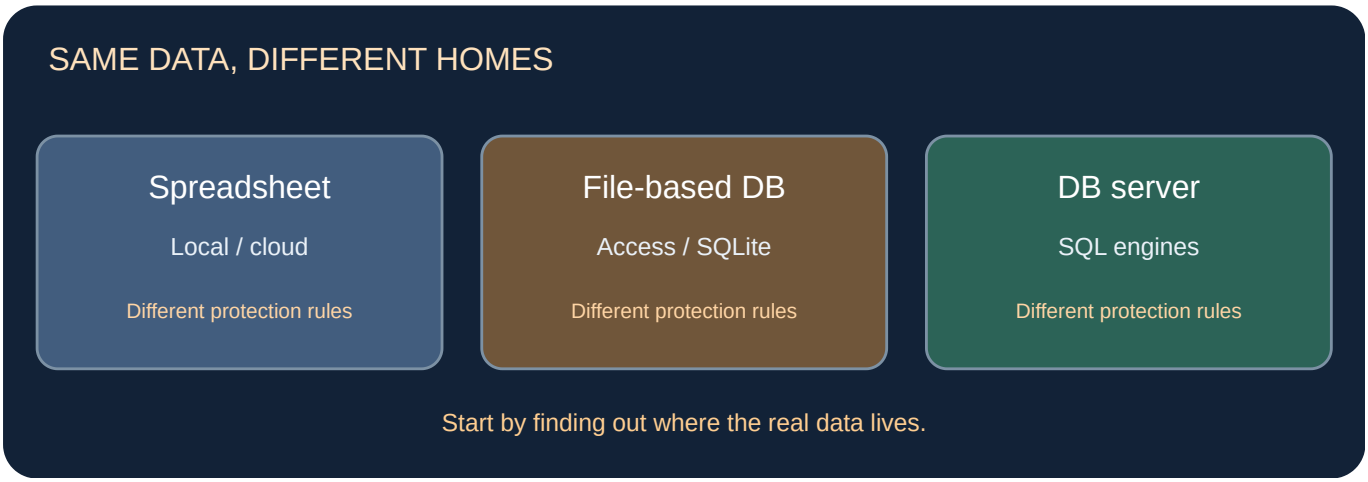
Chapter two: you can have a database without realising it

— AGNIESZKA! WE HAVE ONE DATABASE!
— And the accounting system?
— ONE!
— The holiday register in Excel?
— THAT'S JUST EXCEL!
— The rota in Google Sheets?
— THAT'S THE CLOUD!
— The fifteen-year-old warehouse software?
— THAT'S A HISTORICAL MONUMENT!
— They're all storing data, boss — says Grzesiu.
— THEN RENAME EVERYTHING “DATABASE”! PROBLEM SOLVED!

🤣🤣

Companies use **server database engines** such as MySQL, MariaDB, PostgreSQL, Microsoft SQL Server and Oracle; **embedded or file-based databases** such as SQLite and Access; older systems such as Pervasive/Btrieve (also seen under the Actian Zen name); plus **spreadsheets and registers** in Excel, LibreOffice Calc and Google Sheets. Other models include document databases such as MongoDB and key–value systems such as Redis, whose persistence and use cases vary.

This isn't a ranking of “good” and “bad”. **SQLite is a genuine relational SQL database** despite not needing a separate server process. Access supports relationships and forms; Pervasive is not merely a folder of passive files in every configuration. A spreadsheet can also be excellent for a small register. Problems begin when nobody knows where the data lives, who can access it and what happens after a failure.



Chapter three: move the app, leave the invoices behind

— AGNIESZKA! I BOUGHT YOU A NEW COMPUTER!
— How kind.
— AND MOVED THE ACCOUNTING PROGRAM'S ICON ONTO IT!
— Did you move the database?
— THE ICON IS RIGHT THERE!
— An icon isn't a database, boss — says Grzesiu.

— THEN WHAT IS IT DOING THERE?!
— Launching software that connects to the real data.
— DON'T TELL ME MY ICON DOESN'T KNOW ANYTHING!

🤣🤣

The same application can use different storage arrangements. In some versions and configurations of accounting software — the Polish **Płatnik** payroll and social-insurance program is one example — local Access-based setups and Microsoft SQL Server installations have been supported. Another app may keep SQLite inside a user profile, while another connects to PostgreSQL or MySQL. Always check the documentation for the particular software version and setup.

Moving an application doesn't necessarily move its data. You may also need database files, logs, connection settings, **certificates**, licences and a compatible environment. Before replacing a PC or reinstalling software, find the actual data store and test the migration and recovery plan.

Chapter four: a sticky note on the monitor? Never!

Grzesiu walks into the accounting office.

— Agnieszka, I'm glad you don't stick passwords to your monitor.
— Of course not! That's unprofessional!
— Excellent. Where did you say you keep them, then?
— Under my socks.
— Alongside the **backup** drive?
— No, Grzesiu. In the other drawer!
— ...
— You said to keep them separate!
— And protected from unauthorised access.
— Nobody opens that drawer. Not even the boss.

🤣🤣🤣

A password on paper **isn't automatically a disaster** if it is properly protected, long and unique. But a visible note stuck to an office monitor or a password stored in a publicly shared spreadsheet creates obvious risks. A password manager, individual user accounts, MFA where supported and an independent emergency access procedure can all help.

The bigger question is **whether an unauthorised person could gain access to the account and the entire database**. Access to an accounting application should not automatically confer full administrative privileges in the database engine.

Chapter five: the manager borrows Agnieszka's password

— AGNIESZKA! I NEED TO SEE AN INVOICE IMMEDIATELY!
— I'll show you, boss.
— NO TIME! GIVE ME YOUR PASSWORD!
— But Grzesiu told me not to.
— I'M THE MANAGER!
— Well, all right...

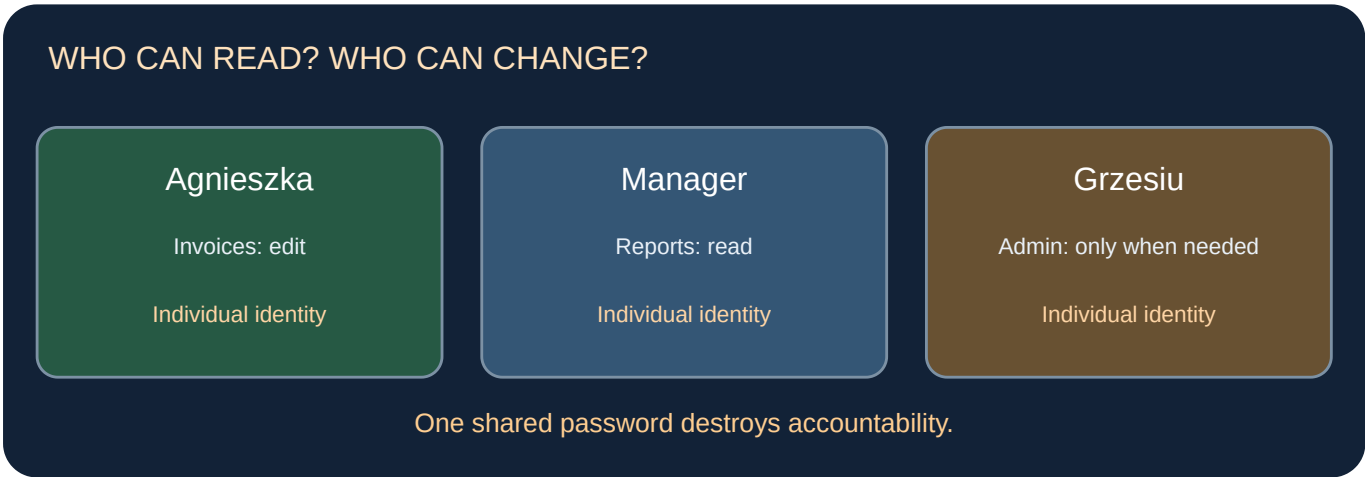
Two hours later.
— GRZESIU! WHO DELETED THREE HUNDRED INVOICES?!
— The audit trail shows Agnieszka's account.
— AGNIESZKA!!!
— But I was working in the other room!
— Boss, who borrowed her password this morning?
— GRZESIU, STOP CHANGING THE SUBJECT!

🤣🤣🤣

Every person should have **their own identity and permissions suited to their work**. A manager who only needs reports doesn't need permission to delete invoices or administer the database. An accountant can edit accounts without becoming a server superuser. Even technical administrators should elevate their privileges only when needed.

One shared password across a department makes accountability difficult. And if the application connects to the database using one service account, the database engine may only record a name such as `app_user`. Proper attribution then depends on reliable application logs and suitably designed auditing.

— GRZESIU! ALL THE CHANGES WERE MADE BY `app_user`!
— That's the application's service account.
— THEN FIRE HIM IMMEDIATELY!



Chapter six: a Google Sheet and a secret available to anyone with the link

— GRZESIU! — calls Agnieszka. — I'VE MADE A REPORT FOR THE BOSS!
— Where?
— Google Sheets. It's wonderful because I can use it on my phone!
— Who has access?
— Anyone with the link.
— Does it include employees' private details?
— Well yes, the manager asked for them.
— Where is the link?
— I sent it to everyone. Don't worry, though, it's a very long link!
— ...
— Is that good?
— Let's restrict access first.

🤣🤣🤣

Google Sheets, cloud-based Excel and shared documents are not inherently unsafe. They can support collaboration, account controls and version history very well. But convenience should never be mistaken for complete control over confidential information. “Anyone with the link” is a different audience from “these specific authorised people”.

Don't thoughtlessly put **passwords, tokens, API keys, identity-document copies, medical records, complete personnel records or sensitive financial information** into broadly shared spreadsheets. This isn't an absolute ban on a provider: assess the service, contractual and legal requirements, settings, access controls and retention. Sometimes the right answer is a smaller dataset or an aggregated report rather than a complete list of individuals.

Chapter seven: the little spreadsheet becomes mission-critical

— AGNIESZKA! WHY CAN'T EVERYONE EDIT THIS EXCEL FILE AT ONCE?!
— Yesterday I changed a customer's address, and you saved an older copy over it.
— THEN DON'T EDIT WHILE I'M EDITING!
— Would you like to prepare an editing timetable?
— THIS IS SUPPOSED TO BE A MODERN SYSTEM!
— Perhaps Grzesiu could check whether our software supports co-authoring and version history?
— GRZESIU! MAKE EXCEL INTO A DATABASE, BUT MAKE IT FREE!

🤣🤣

A spreadsheet containing a few dozen records can be an excellent tool. Trouble starts when it becomes the central system for an entire company: many editors change the same entries, formulas go wrong, permissions apply to the whole file, and operations aren't transactional. Modern spreadsheet services support collaborative editing and version histories, but **that doesn't automatically provide the same integrity model as a properly designed transactional database**.

Distinguish an ordinary shared file on a network drive, a cloud spreadsheet with co-authoring and an application backed by a database server. Not every spreadsheet is bad. **The real danger is ignoring the limitations of the tool entrusted with important information.**

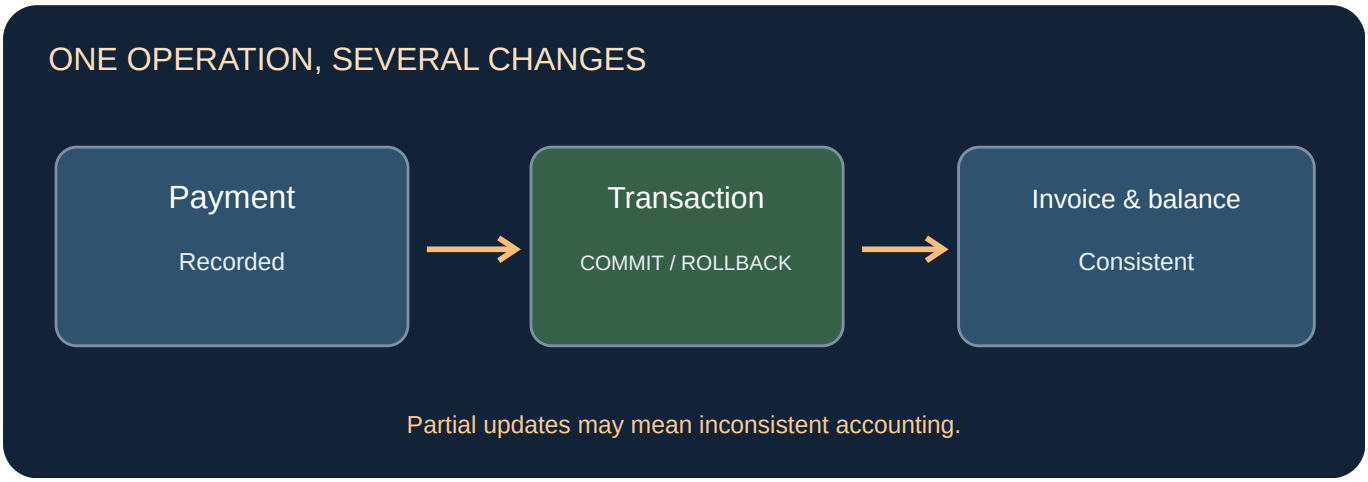
Chapter eight: one payment, three numbers and a power cut

— GRZESIU! — says Agnieszka. — THE CUSTOMER PAID, BUT THE COMPUTER STILL SAYS THEY OWE US!
— Is the payment recorded?
— Yes.
— And the balance?
— The old figure.
— And the invoice status?
— “Unpaid”.
— THEN TELL THE CUSTOMER TO PAY AGAIN! — orders the manager.
— Boss, they already paid. We need to fix the data, not the customer's wallet.
— WHY CAN'T THE COMPUTER ADD NUMBERS PROPERLY?!

🤣🤣

Transactional databases can group related updates. **COMMIT** confirms a successful transaction, while **ROLLBACK** reverses changes in an uncommitted transaction; it isn't a universal undo button for anything that happened last month. The **ACID** principles describe transaction properties, including atomicity and durability, with details depending on the engine and its configuration.

Integrity constraints may ensure, for instance, that a referenced customer exists or a document identifier is unique. They don't replace correct accounting rules or application logic. A database can be structurally consistent yet contain the wrong amount because someone entered it incorrectly.



Chapter nine: copying a live database folder

— AGNIESZKA, DID YOU MAKE A **BACKUP**? — asks Grzesiu.
— Of course! I copied the program folder to a USB drive.
— While you were working?
— Yes, so I wouldn't interrupt the accounts.
— Is the database file-based or server-based?
— It has a green icon, Grzesiu. I truly don't know.
— Let's check whether the application has a **backup** function, or whether the database engine needs a specific method.
— So I shouldn't throw away the old drive yet?
— Agnieszka, that's an excellent idea.

🤣🤣

Ordinary file copying while a database is running may not produce a usable backup. Database engines may keep changes in memory and log files, lock files or require a consistent point-in-time snapshot. SQLite can use separate WAL files, Access has its own shared-access rules, while MySQL, PostgreSQL and SQL Server have engine-specific **backup** mechanisms.

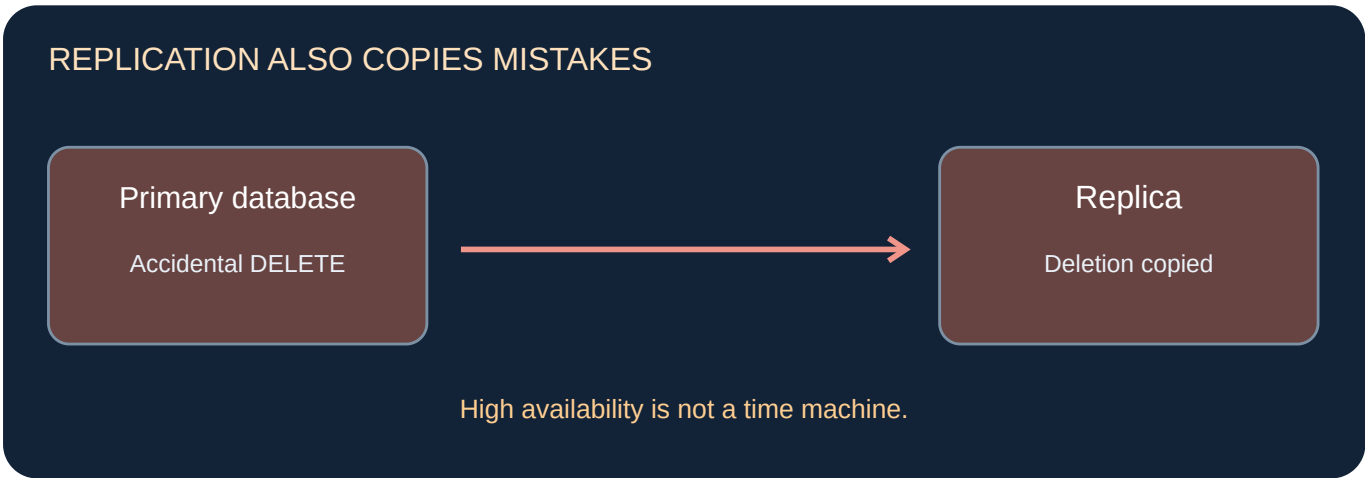
The right method depends on the app and its setup. Sometimes it's a logical **backup**, sometimes a physical one, an application-consistent snapshot or a carefully managed shutdown. **Never assume that copying a random folder means you can restore the database.**

Chapter ten: the replica is very obedient

— GRZESIU! WE HAVE A SECOND DATABASE SERVER!
— Yes, it's a replica.
— SO NOTHING CAN EVER DISAPPEAR!
— Replicas help with some failures and workloads, but they aren't historical **backups**.
— WHAT IF I DELETE THE INVOICES?!
— The deletion can be replicated.
— ON THE SECOND SERVER TOO?!
— Yes, that's what replication does.
— WHY DID I BUY A SECOND SERVER IF IT REPEATS THE FIRST ONE'S MISTAKES?!

🤣🤣

Replication and high availability aren't the same as backups. A second server may quickly take over after hardware failure, while also copying mistaken commands, logical corruption or unwanted changes. Historical recovery points, retention and independent permissions provide a different kind of protection.



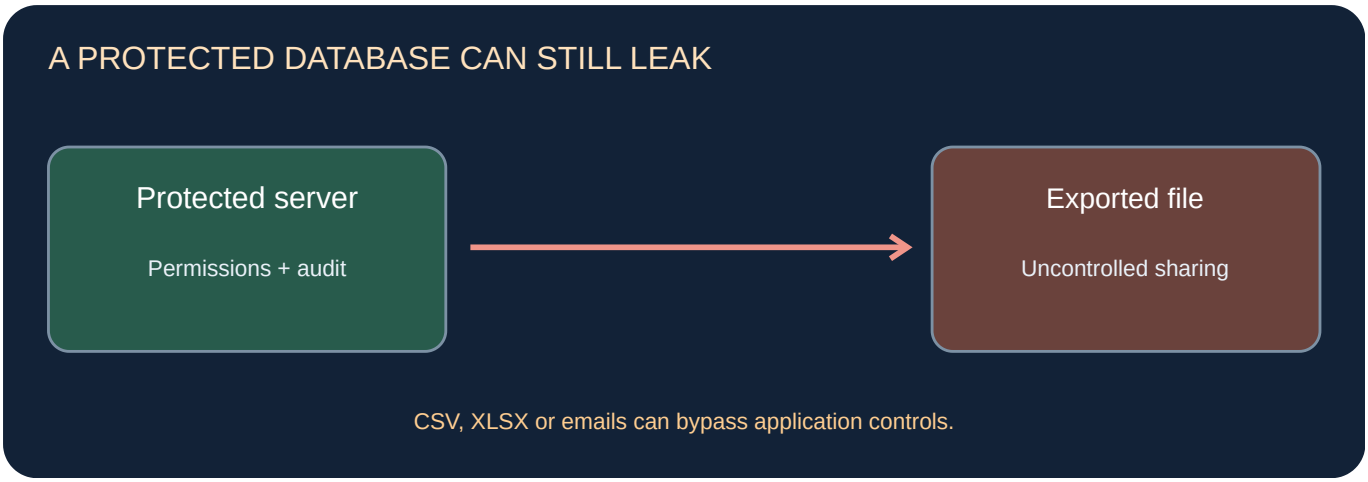
Chapter eleven: secrets should not travel with exports

— AGNIESZKA, WHERE IS MY REPORT?! — asks the manager.
— I emailed it to you.
— EXCELLENT!
— Where did the numbers come from? — asks Grzesiu.
— I exported the entire database so I wouldn't miss anything.
— The entire database?
— Yes. Then I deleted the unnecessary columns in my Excel file.
— But you sent the original export first?
— ...
— ...
— GRZESIU, IS THERE A WAY TO UNDO THE INTERNET?!

🤣🤣

Even a well-protected database can leak through **uncontrolled exports to CSV, XLSX, PDF, email attachments or personal cloud storage**. Server permissions and at-rest **encryption** don't automatically protect a file that someone downloads and sends elsewhere.

Limit exports to required fields, check recipients, protect files, apply sensible retention and define who may export data. An exported report may be just as confidential as its source database.



Chapter twelve: not everything you can collect should be kept

— AGNIESZKA! WE'VE KEPT EVERY CUSTOMER RECORD SINCE 1997!
— What for, boss?
— IT MIGHT BE USEFUL!
— Useful for what?
— ...
— Boss?
— STOP ASKING QUESTIONS I DON'T HAVE A FORM FOR!

🤣🤣

An important protection principle is **data minimisation**: collect and keep what is genuinely needed for a defined purpose. That doesn't mean automatically deleting old accounting records. Some document types must legally be kept for years, and requirements vary by jurisdiction and record type.

But a private copy of the entire customer database “just in case” isn't safe because nobody remembers it exists. Know where exports live, who can access them, when deletion is appropriate and how **backup** retention works.

Chapter thirteen: the update and a database nobody can open

— AGNIESZKA! I'VE UPGRADED THE ACCOUNTING SOFTWARE!
— Did you back up the database first?
— THE SOFTWARE WAS NEW!
— And the data?
— OLD!
— Did the upgrade change its format?
— IT SAYS “CONVERSION COMPLETE”!
— And the old software can't read the converted database — says Grzesiu.
— THEN UNINSTALL THE CONVERSION!
— That's not a separate application.
— THEN FILE A COMPLAINT WITH THE FUTURE!

🤣🤣

Software upgrades can change database schemas, file formats and environment requirements. **Reinstalling the previous app doesn't necessarily reverse the data migration.** Before major changes, check compatibility, make a suitable **backup**, test recovery and understand how a rollback would actually work.

Long-lived archives have another risk: a decade from now you may lack compatible software, drivers, licences or operating systems. Plan for **long-term readability**, including documented exports to suitable formats and a way to run required legacy environments — without assuming a plain CSV captures every feature of a database.

Chapter fourteen: Agnieszka's extraordinary professional luck

The accounting office erupts in alarm.

— GRZESIU! I CAN'T OPEN THE ACCOUNTS!
— What happened?
— The manager moved the program, upgraded the computer and said we no longer need the old server.
— Did you change anything yourself?
— No. I remembered what you said: “If you don't know, don't click.”
— And the **backup**?
— There's the one you make. And I have a note with your phone number.
— Excellent. Where's the note?
— In the drawer.
— Under the socks?
— No. In the other one!

🤣🤣🤣

Agnieszka doesn't have to repair SQL servers, understand database log files or explain the difference between SQLite and PostgreSQL. She needs to know when to stop work, how to report trouble, whom to call and why she shouldn't experiment on live data. **Knowing your professional limits is a strength, not a weakness.**

Grzesiu's job is to verify **backups**, limit permissions, choose a safe **recovery** procedure and document the setup. Agnieszka's luck is charming, but **luck is not a substitute for planned protection**.

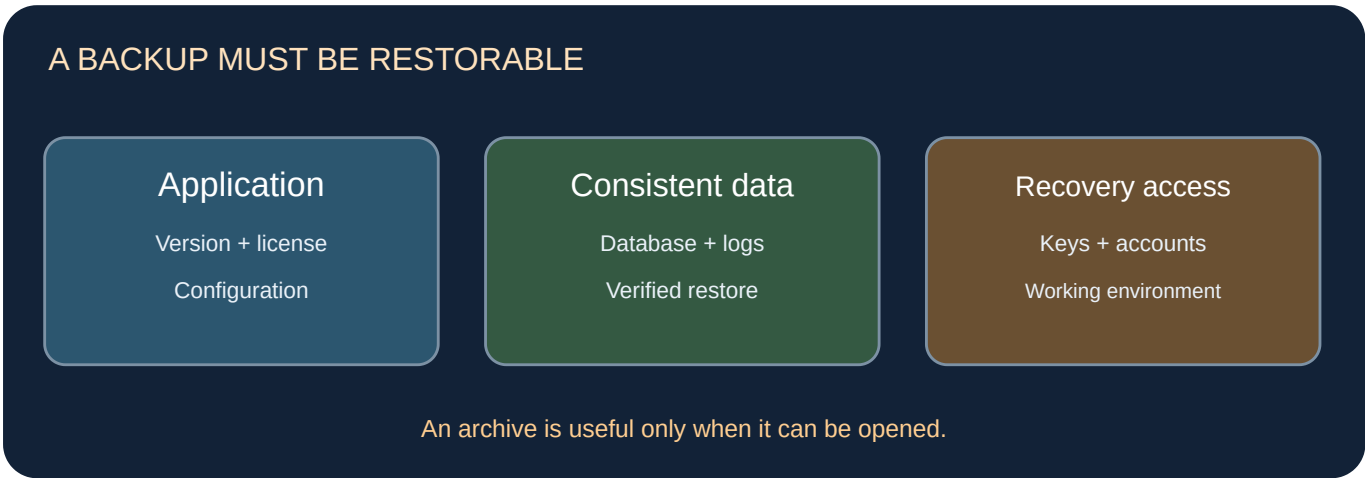
Chapter fifteen: can we really get back to work?

— GRZESIU! WE HAVE A DATABASE **BACKUP**! — cheers the manager.
— Good. How will we restore it?
— DOUBLE-CLICK IT!
— What if we need the older application version, a licence, configuration and keys?
— ...
— What if the **backup** needs transaction logs?
— ...
— Boss?
— AGNIESZKA! WHERE ARE THOSE SOCKS OF YOURS?!

🤣🤣

A valid database **backup** isn't always one file. You may need the correct database engine or application, a compatible schema version, accounts, configuration, logs, keys and a verified method of starting the system. Some setups can recover to a chosen moment in time — **point-in-time recovery (PITR)** — if the required **backups** and logs have been retained.

Run periodic recovery tests: check that records are complete, accounts reconcile, user permissions are appropriate and the application genuinely works. **A database backup that cannot restore useful business operations is only half a rescue plan.**



Finale: the socks are promoted to the security department

Month end. Agnieszka finishes the accounts. Grzesiu confirms that **backups** have run and recovery has been tested. The manager storms in, triumphant.

— AGNIESZKA! WE HAVE A PROFESSIONAL DATABASE SERVER!
— That's wonderful.
— INDIVIDUAL USER ACCOUNTS!
— Excellent.
— AN AUDIT TRAIL!
— Lovely.
— **BACKUPS** AND RESTORE TESTS!
— Grzesiu has worked hard.
— SO YOU CAN THROW AWAY THE OLD DRIVE UNDER YOUR SOCKS!
— No, boss.
— WHY NOT?!
— Because I like having more than one way out. And Grzesiu has a plan for when we replace that drive.
— AGNIESZKA, YOU DON'T RESPECT ME AS A MANAGER!
— I do. I just respect the data a little more.

🤣🤣🤣🤣

Remember: database security isn't determined solely by whether you use MySQL, PostgreSQL, MSSQL, Access, SQLite, Pervasive, Excel or Google Sheets. It depends on genuine access restrictions, data consistency, controlled sharing, correct **backups** and successful restoration. **The best system protects the information, not merely the appearance of professionalism.**

*Accountant Agnieszka returned to her invoices. Grzesiu returned to his servers. Frank, reassigned to transport after the sign-on-the-car incident, received fresh instructions: never label transported **backup** drives in enormous letters. The manager ordered a rubber stamp reading “DATABASE SECURED”.*