Wevo GL Analytics

Business Guide

Implementing Multi-Level Management Reporting

Document reference: WEVO-OPS-GUI-001

Document control

Field	Detail
Document title	Wevo GL Analytics Business Guide - Implementing Multi-Level Management Reporting
Document reference	WEVO-OPS-GUI-001
Version	5.0
Status	Issued
Audience	Finance managers, financial controllers, accountants and adoption teams
Classification	Distribute freely

Contents

1 Purpose

This guide explains how an organisation adopts multi-level management reporting using Wevo GL Analytics: how the codes already present in the general ledger become the leaves of management hierarchies, how those hierarchies are designed, built and validated, how they are paired into cubes, how GL data flows in to keep the cubes current, and how the business then uses the result. It is written for the people who will make the adoption happen - finance managers, financial controllers, accountants and the adoption team - and it assumes no technical knowledge.

The guide is deliberately practical. Alongside the concepts, it follows a single worked example from first chapter to last: a banking group whose general ledger, MasterGL, feeds Wevo. The group designs a cost centre hierarchy that reflects its management structure, pairs it with its chart of accounts, and ends with a live, drillable view of its figures. Every screen image in this guide is taken from that example, so what is described and what is shown always match.

1.1 Where this guide sits

This is the second document in the Wevo GL Analytics document set. It bridges the Business Introduction, which explains what Wevo is and the value it delivers, and the two user manuals, which document every screen control by control. Read this guide to understand how to do management reporting with Wevo; turn to the manuals when you need the mechanics of a particular screen.

Document	Reference	Answers the question
Business Introduction	Document 01	What is Wevo GL Analytics, and why adopt it?
Implementing Multi-Level Management Reporting (this guide)	WEVO-OPS-GUI-001	How do we take our GL and turn it into management reporting?
User Manual - Hierarchy and Cube Development	WEVO-ENG-GUI-001	How does each maintenance screen work, field by field?
User Manual - Cube Drill-Down Enquiry	WEVO-ENG-GUI-002	How does each enquiry screen work, field by field?

1.2 How to read this guide

The chapters follow the order of adoption: understand the reporting model, know your GL codes, design the hierarchies, build and validate them, create the cubes, establish the data feed, use the reporting, and then operate and evolve it. Each chapter opens with the business concept, shows how Wevo performs the function, carries the worked example forward, and closes with a pointer to the manual section that documents the screens involved. A reader who works through the chapters in order will finish with a complete picture of the adoption; a reader with a specific question can go straight to the relevant chapter.

2 Management reporting - the concept

A general ledger stores the financial truth of an organisation as flat, linear entries: a code combination, a period, an amount. That format is ideal for posting and audit, and almost useless for management. A divisional director does not think in terms of ten thousand cost centre and account code combinations; they think in terms of their division, the departments within it, and the responsibilities each department carries. Management reporting - historically called responsibility reporting - is the discipline of presenting GL data through the structures by which the business is actually managed and operated, so that every figure lands with the person accountable for it.

The mechanism is aggregation through a hierarchy. Each GL code becomes the bottom of a tree; above it sits groupings that mirror the management structure; at the top sits a single consolidated node. Post a journal to one cost centre and its effect rolls up, level by level, to every summary that cost centre belongs to. Reporting then becomes navigation: start at the consolidated figure, and drill down through the same tree until the detail that explains it is on screen.

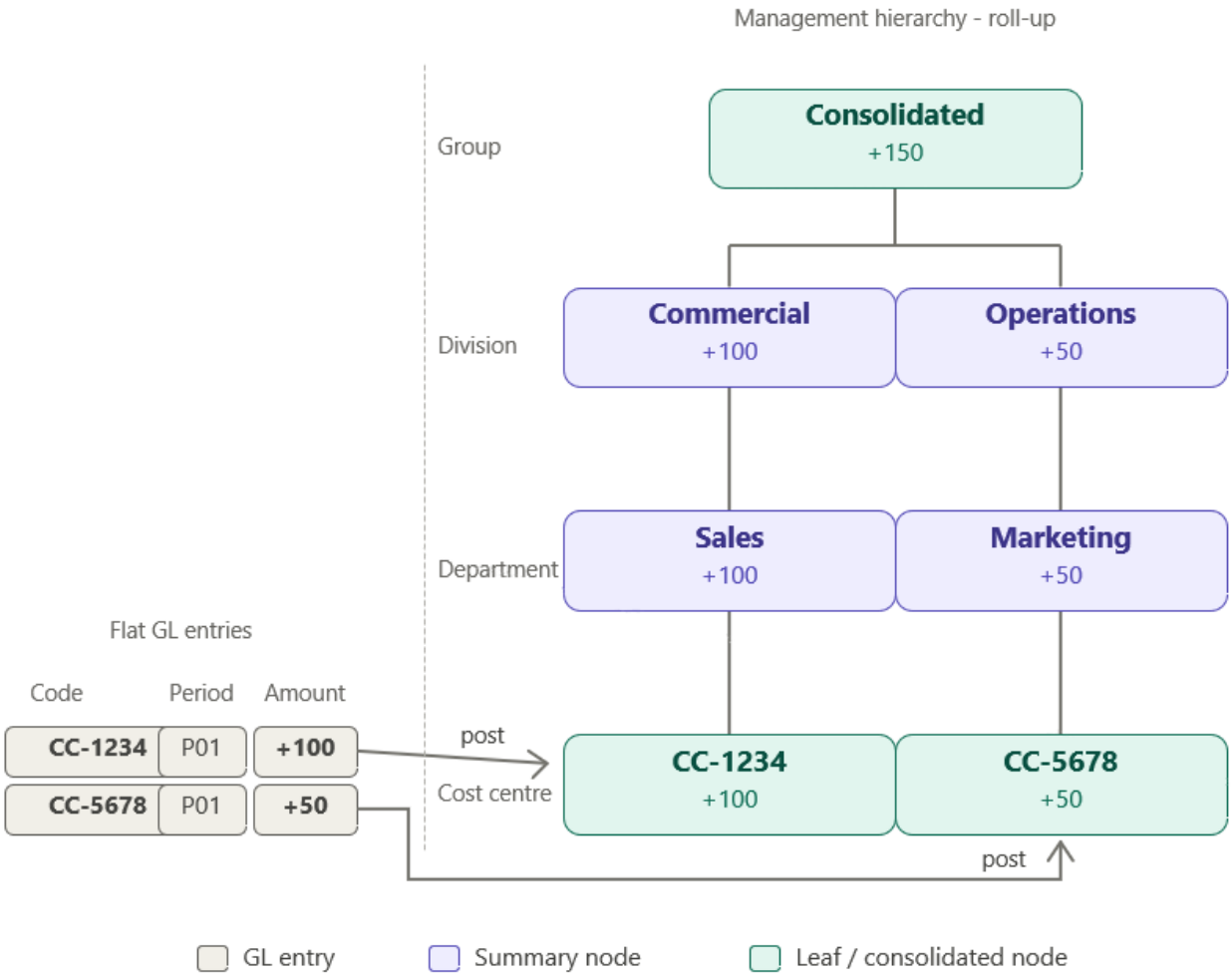

Figure 1 - Flat GL entries summarised through a management hierarchy to the consolidated view.

Three properties make this model powerful in practice. First, ownership: every node in the hierarchy has a natural owner, so accountability is built into the structure of the reporting rather than bolted on. Second, consistency: because every view is produced from the same tree over the same data, a figure at any level always reconciles exactly to the detail beneath it. Third, flexibility: the business can define as many hierarchies as it has ways of looking at itself - management structure, legal

entity, product line, geography - all over the same underlying GL data, without touching the source system.

2.1 What Wevo brings to the model

Management reporting of this kind is a concept; Wevo GL Analytics is its implementation. Wevo holds the hierarchies, receives the GL data, performs every roll-up in advance, and serves the result through a web interface that is typically minutes behind the ledger. Because every aggregation is pre-calculated and stored, a drill from the group total to an individual cost centre returns in well under a second, at any time, with no computation against the source GL. The GL system remains the authoritative record throughout; Wevo is a read-only analytical layer on top of it.

2.2 The worked example

Throughout this guide we follow a banking group adopting Wevo. The group runs a general ledger known as MasterGL, whose accounting key carries three segments: Business Unit, Cost Centre and Account Code. Management wants a reporting view that starts at the Global Banking Group, breaks into divisions such as Corporate Banking and Legal and Governance, then departments, sub-departments, and finally the individual cost centres where costs are actually posted - branches, call centres, ATM networks and processing centres. On the other axis, the group chart of accounts needs the same treatment: fee income, commission income, advisory service fees and so on, grouped for management rather than for posting. By the final chapter, both structures are live in Wevo and the group is drilling from consolidated figures to individual postings.

3 Knowing your GL - segment values

Every adoption starts in the same place: the codes the general ledger already uses. Each entry in a GL is classified by an accounting key made up of segments - typically two to four of them - and each segment carries a set of valid codes. One segment might carry business unit codes, another cost centre codes, another account codes. These individual codes are the segment values, and they matter because they are the raw material of every hierarchy: each leaf node of a Wevo hierarchy is one real segment value, named identically to the code in the GL, with no transformation or aliasing.

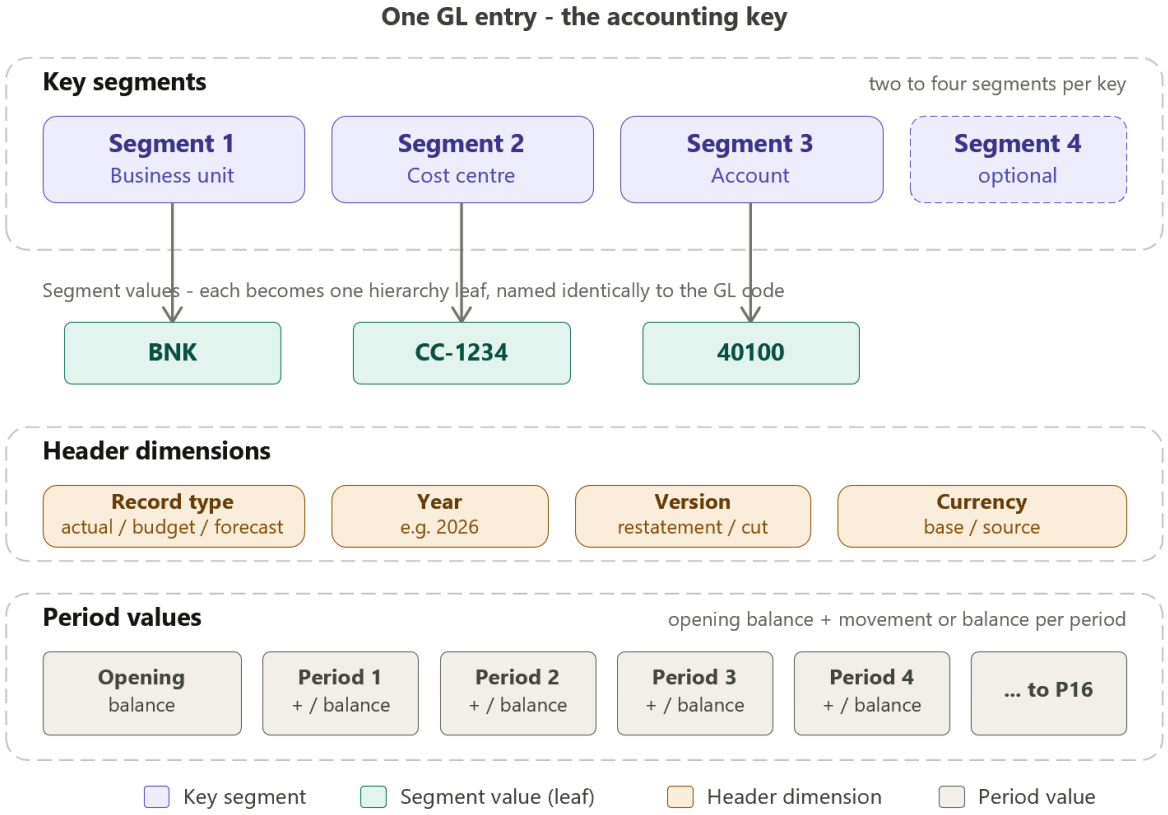

Figure 2 - The general ledger accounting key - segments, record type, year, version, currency and period values.

Alongside the key segments, each GL entry carries a record type (actuals, budgets, forecasts), a year, a version, a currency, and the period values themselves - an opening balance plus the movements or balances for each accounting period. Wevo carries all of these dimensions through to the reporting, which is why the same hierarchy can present actuals against budget, this year against last, and base currency against source currency without any additional structure.

3.1 One ledger or several

Segment values arrive in Wevo through the same file interface as the financial data, and each value is tagged with the source system it came from. An organisation with a single GL will see one system name against every value; a group that runs several ledgers - after an acquisition, for example, or across regions - will see each system's values side by side. This matters at design time: where two ledgers use different code ranges for the same real-world things, the hierarchy is the place they are brought together, because leaf nodes from any source system can sit under the same summary structure.

3.2 How Wevo presents segment values

The Segment Values screen in the Hierarchy and Cube Development application is the reference view of everything the GL has supplied. One tab per segment - the tab labels are driven by system parameters, so they carry your organisation's own terminology - with a keyword search and a source system filter. The screen is deliberately read-only: segment values are owned by the GL, defined and maintained there, and Wevo simply reflects them.

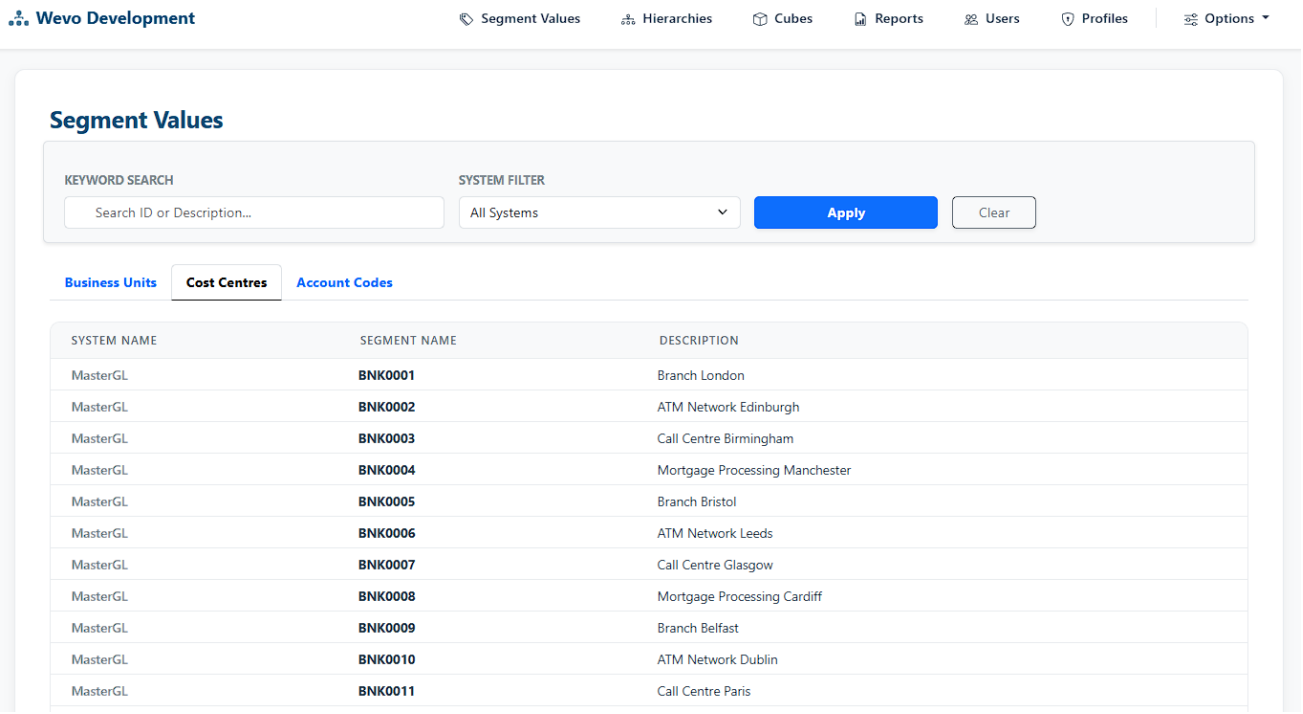

Figure 3 - The Segment Values screen - the read-only reference view of the codes supplied by the GL.

3.3 The worked example - MasterGL's codes

Our banking group's ledger, MasterGL, supplies values for all three segments. The cost centre segment carries codes such as BNK0001 Branch London, BNK0002 ATM Network Edinburgh, BNK0003 Call Centre Birmingham and several thousand more; the account code segment carries the group's posting accounts, among them the advisory service fee accounts 4068 and 4069. Before any design work begins, the adoption team reviews these values on the Segment Values screen: which codes are active, what the descriptions say, and whether the descriptions are fit to appear on management reports - because leaf node descriptions are what business readers will see at the bottom of every drill.

3.4 What the business should prepare

- An owner for each segment: who in finance answers for the completeness and quality of the business unit, cost centre and account code populations.
- A review of code descriptions: they will surface on reports, so poor or abbreviated descriptions are worth correcting in the source GL before go-live.
- A view on dormant codes: codes that no longer receive postings still appear as available values, and the design should decide where they will sit.

In the User Manual - Hierarchy and Cube Development, see the Segment Values section for the screen reference.

4 Designing hierarchies

A hierarchy is a tree of named nodes built on one GL segment. At the bottom sit the leaf nodes - the real GL segment values. Above them sit summary nodes, which exist only in Wevo and only for reporting: they are the divisions, departments and groupings through which management wants to see the figures. Every node except the root has exactly one parent, and the root sits alone at the top as the consolidated view. Levels give the tree its shape: each level is a named tier - Division, Department, Sub-department - and every node belongs to exactly one level.

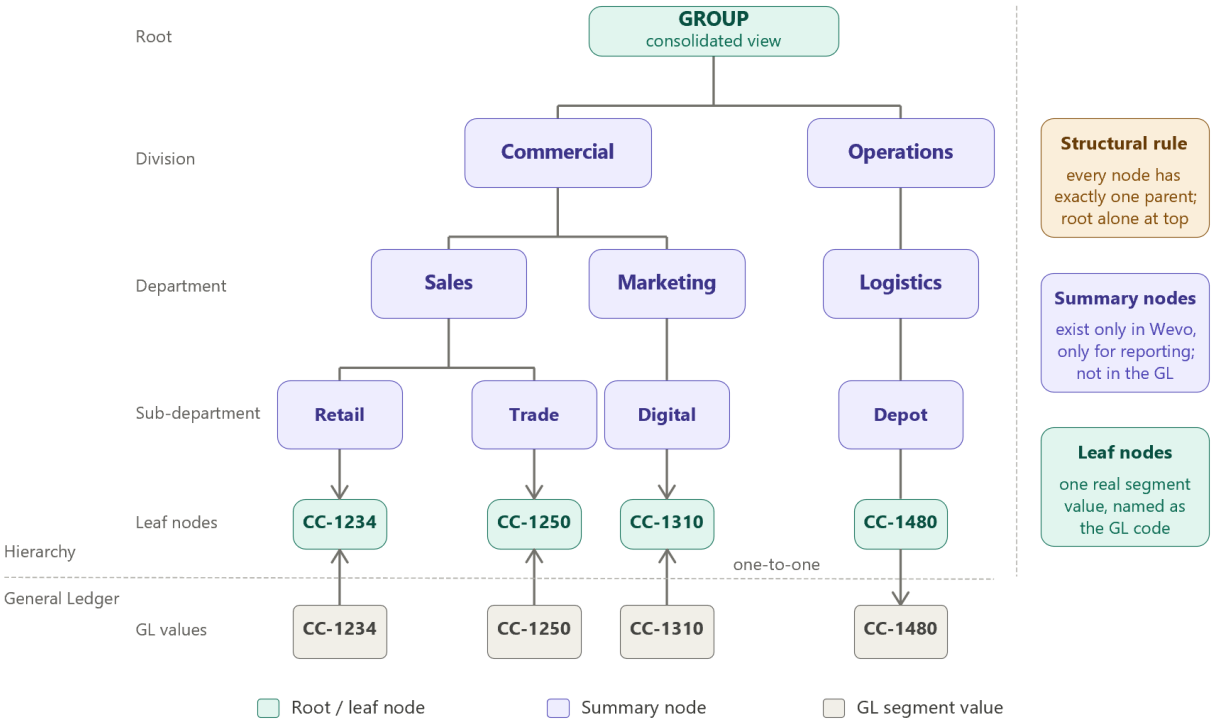

Figure 4 - Anatomy of a hierarchy - the root, summary levels, and leaf nodes mapped one-to-one to GL segment values.

Designing a hierarchy is a business exercise, not a technical one. The question it answers is: through what structure should accountability for these figures flow? For an organisational hierarchy the answer usually already exists - it is the management structure, and the design work is to formalise it. For a chart of accounts hierarchy the answer is the grouping that makes management sense of the accounts: income lines, cost lines, and the sub-analyses each requires.

4.1 Design decisions

Decision	What to consider
Number of levels	Enough tiers that each drill step is meaningful, few enough that a drill from top to bottom is quick. Four to six levels between root and leaf suits most organisations.
Level names	Business terms - Division, Department, Region - not codes. Level names appear throughout the enquiry screens.

Node naming convention	Node names are limited to 15 characters and descriptions to 75. Agree a convention before building: consistent prefixes and numbering make thousands of nodes manageable.
Complete or partial	A complete hierarchy must account for every segment value in the segment; a partial hierarchy covers a chosen subset. Organisational hierarchies are usually complete; analytical side-views are often partial.
Ownership and sign-off	Each hierarchy needs an owner who approves the structure before promotion and adjudicates where a code should sit when departments disagree.
Restructure policy	Reorganisations are certain. Decide early who may restructure, how changes are rehearsed, and when they take effect - Wevo can preserve each year's structure for history, so this is policy rather than constraint.

4.2 Complete and partial hierarchies

A complete hierarchy accounts for every value in its segment: every cost centre, without exception, has a place in the tree. That guarantee is what lets a complete hierarchy reconcile to the GL total - nothing can be posted that does not roll up somewhere. A partial hierarchy deliberately covers a subset: perhaps only the retail network, or only the income accounts, for a focused analytical view. Both are first-class structures in Wevo; the choice is declared when the hierarchy is created, and validation enforces whichever rule applies.

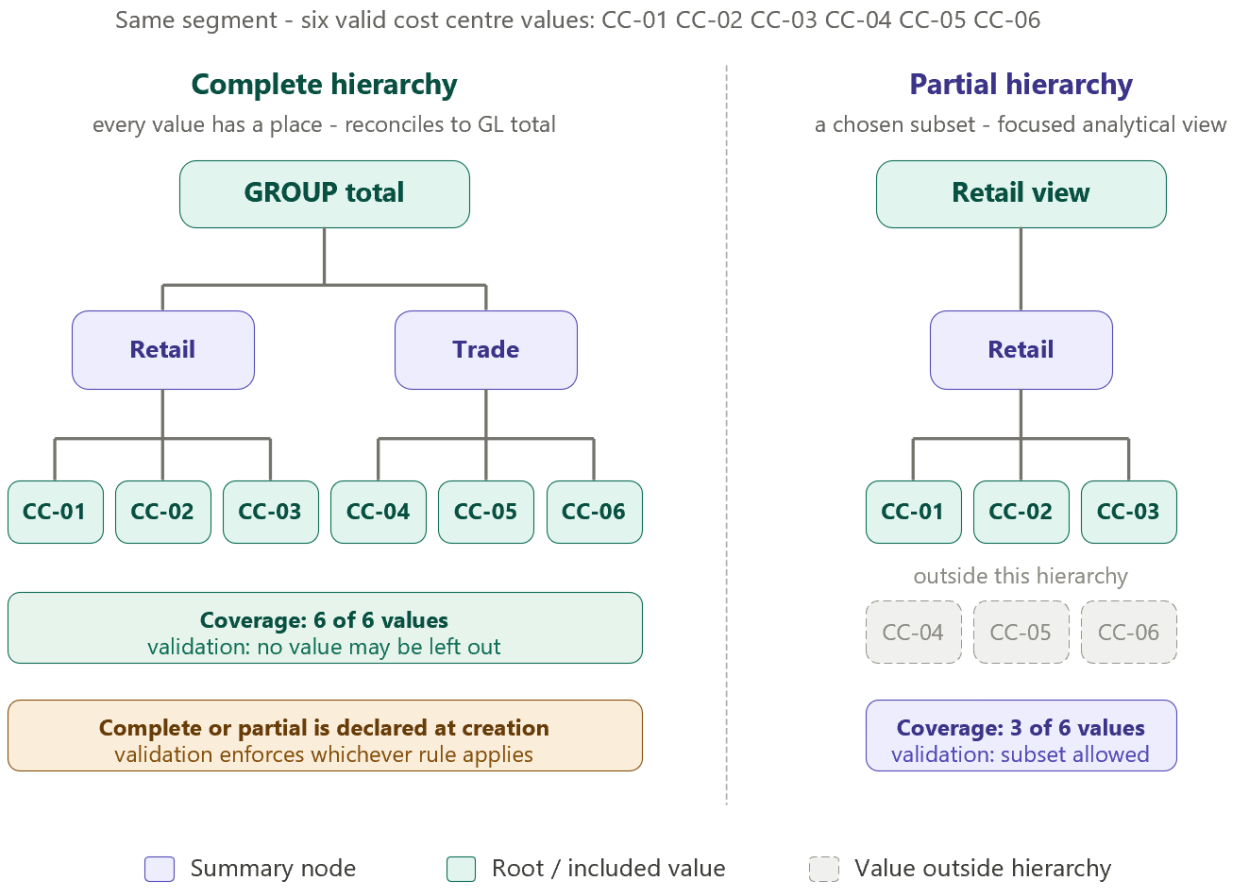

Figure 5 - Complete and partial hierarchies - full segment coverage against a chosen subset.

4.3 The worked example - designing GROUP DIVISION (CC)

The banking group designs its cost centre hierarchy, GROUP DIVISION (CC), as a complete hierarchy on the cost centre segment, with the management structure as its shape. The root is the Global Banking Group. Beneath it, level 200 carries the divisions - Corporate Banking Division, Legal and Governance Division and their peers. Level 300 carries departments such as Legal Operations and the Syndicated Lending Desk; level 400 the sub-departments, which for the international businesses are regions - Asia Pacific, Continental Europe, Latin America; level 500 carries country groupings; and level 900, the leaf level, carries the cost centres themselves - the BNK codes reviewed in the previous chapter. Some three thousand nodes in all, of which around two thousand are leaves.

The same exercise, run over the account code segment, produces the CHART OF ACCOUNTS hierarchy: income and expense groupings at the top, account families such as Fee and Commission Income and Wealth Management Fees in the middle, and the posting accounts as leaves. The group chooses to build this one as a partial hierarchy initially, covering the income statement accounts that management reporting needs first.

The tree below shows the finished structure as Wevo presents it - the Business View of the hierarchy, read in management terms, with each node carrying its child count.

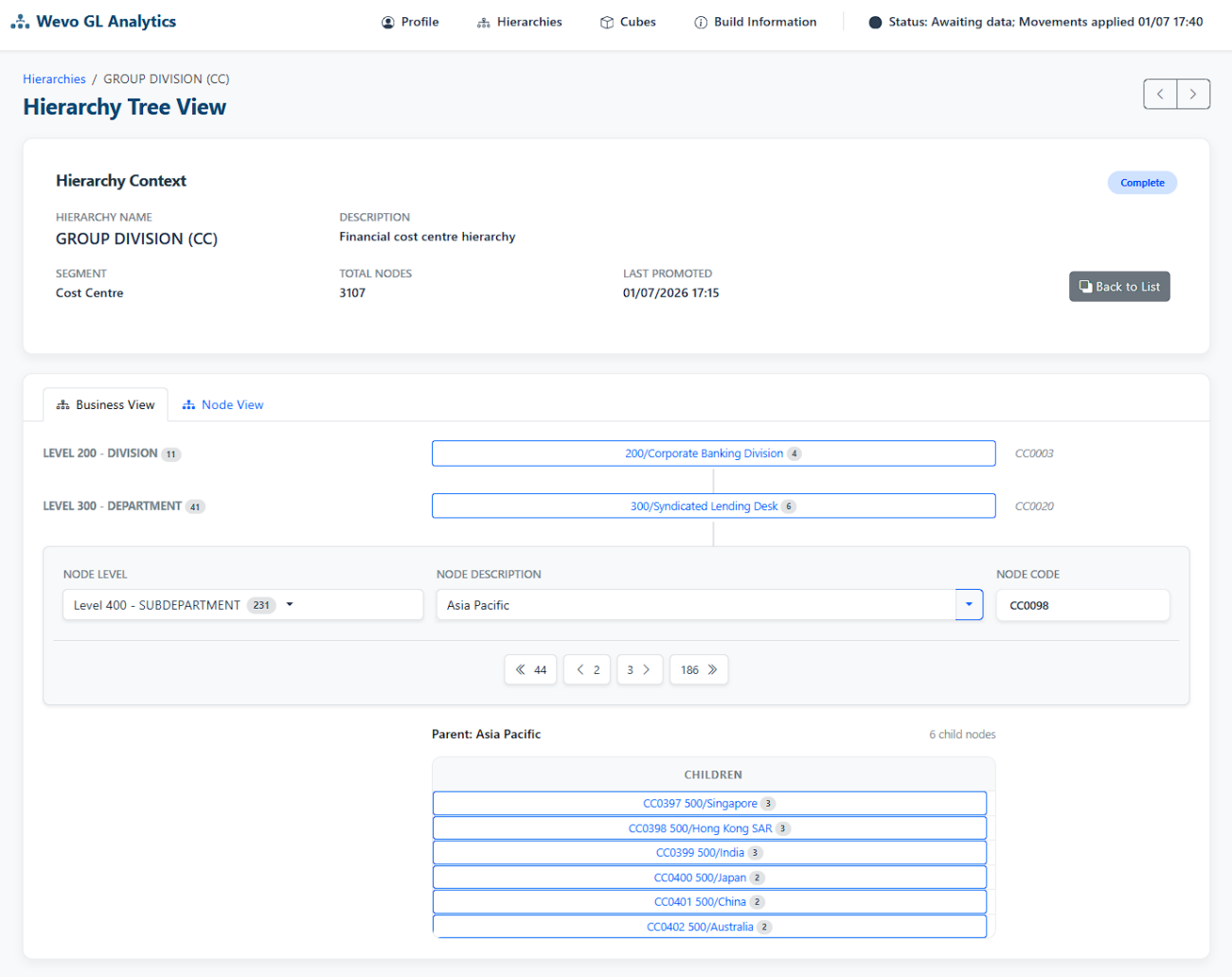

Figure 6 - The designed hierarchy as the business reads it - divisions, departments and regions in the Business View.

5 Building and validating hierarchies

With the design agreed, building is methodical. Everything happens in the Hierarchy and Cube Development application, in a development copy of the hierarchy that has no effect on reporting until it is validated and promoted into production. The sequence is always the same: create the hierarchy, define its levels, add the summary nodes from the top down, assign the segment values as leaves, then validate and promote.

5.1 Creating the hierarchy and its levels

A new hierarchy needs only a name, a description, the segment it is built on, and whether it is complete or partial. On creation Wevo establishes the root and leaf levels automatically and sets the hierarchy's status to NEW. The Manage Levels screen then defines the tiers between: each level has a number, a name and a description, and the numbering convention - 100 for the root, 900 for the leaves, with room between - leaves space for tiers to be added later without renumbering.

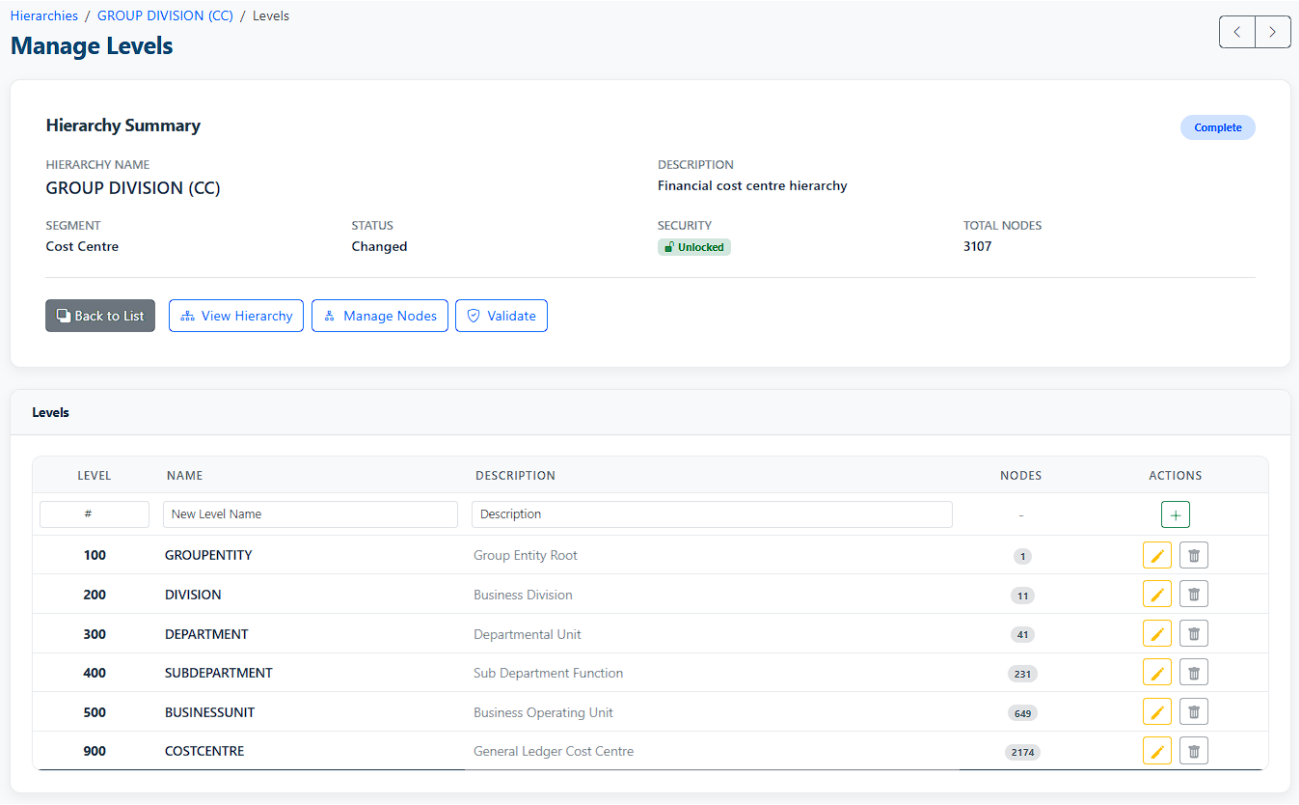

Figure 7 - Manage Levels - defining the named tiers of the hierarchy.

5.2 Adding nodes

Summary nodes are added level by level on the Manage Nodes screen, each with its name, description and parent. The Level View lists every node at a chosen level for bulk work; the Tree View shows any node in the context of its parents and children; the Maintain Node tab handles individual changes - linking and unlinking parents, and promoting or demoting a node to reposition it within the pane so a parent or child can be linked. In practice the top of the tree takes shape quickly - there are few divisions - and the effort concentrates at the leaf level.

Leaf assignment is where the design meets the GL. At the leaf level (say level 900), the Level View offers the Unallocated Segment Values list: every code the GL has supplied that does not yet have a place in this hierarchy, each with a parent selector beside it. Assigning a leaf is one action -

choose the parent, add. For a complete hierarchy this list is also the progress meter: when it is empty, every cost centre has a home.

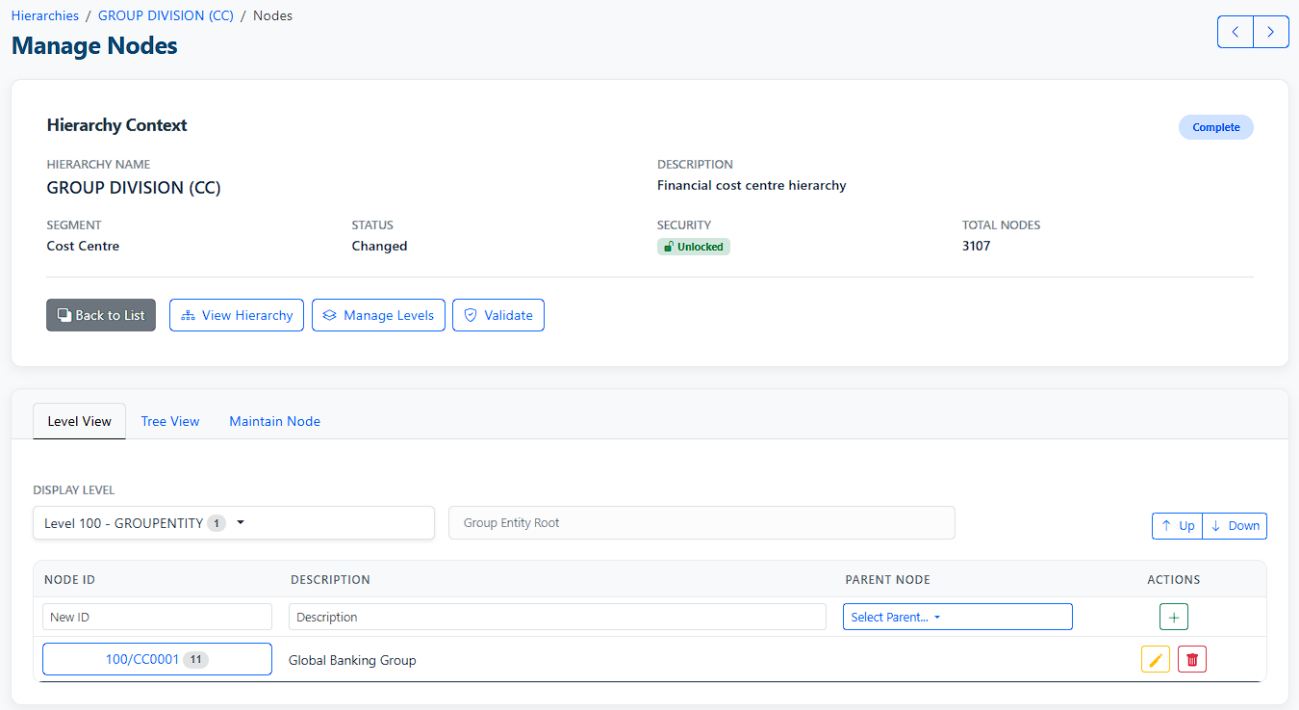

Figure 8 - The Unallocated Segment Values list - assigning GL codes as leaf nodes, one action per code.

5.3 Validation - ensure the hierarchy is structurally correct

A hierarchy cannot reach production reporting until it passes validation. The Validate screen runs the full rule set and reports failures in three groups - level errors, node errors and link errors - each row carrying a description and, where the fix is obvious, a contextual button that jumps straight to the offending item. The checks answer the questions that matter: does every node have a parent, is the tree free of orphans and loops, does every level in use exist, and - for a complete hierarchy - is every segment value assigned as a leaf node. Errors can be exported to a CSV file for offline review and division of labour.

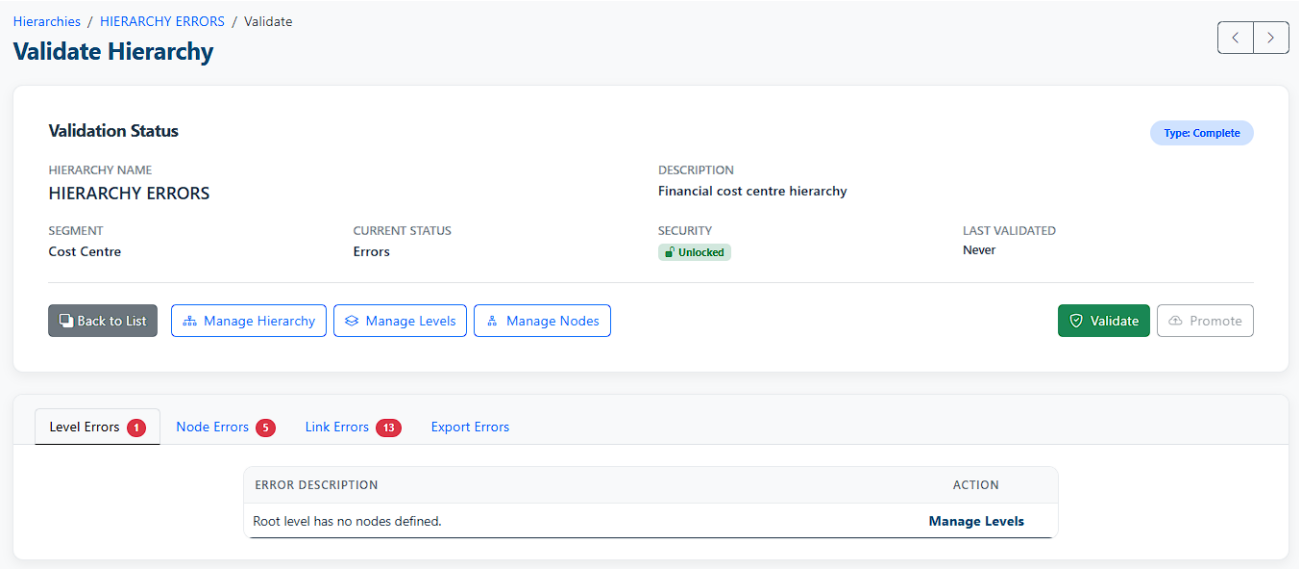

Figure 9 - The Validate screen - the rule set that stands between a draft structure and production reporting.

5.4 The lifecycle

Every hierarchy carries a status that tells the business exactly where it stands. It is born NEW; any edit marks it CHANGED; a clean validation run marks it VALIDATED; and the promote action releases the validated structure to production with a status of PROMOTE. At the start of the next full balance build the promoted hierarchy is copied across to production, where it is then used for summarisation and user navigation. Any subsequent edit in development returns the status to CHANGED, and the cycle repeats - so a hierarchy in production is always one that has passed validation in exactly the form promoted. A separate locked flag lets an administrator freeze a hierarchy in the development environment against any edits, for example during a period close.

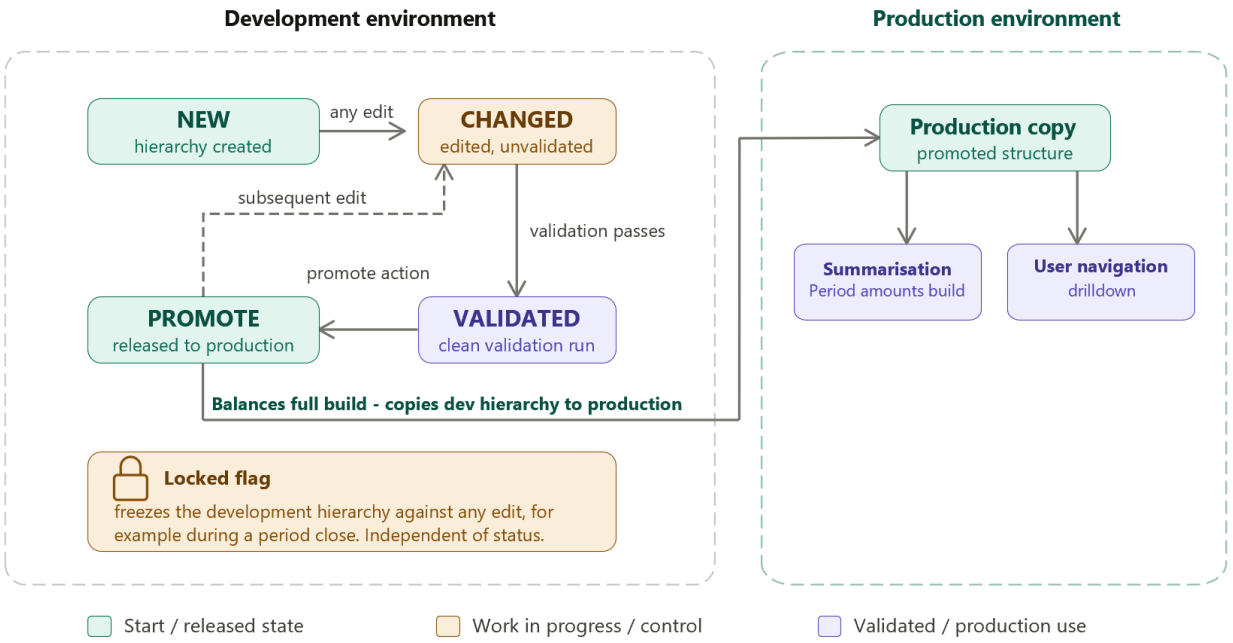

Figure 10 - The hierarchy lifecycle - NEW, CHANGED, VALIDATED and PROMOTE, with development and production copies.

The worked example proceeds exactly this way: GROUP DIVISION (CC) is created as a complete hierarchy on the cost centre segment, its seven levels defined, its divisions and departments added over a working session, and its two thousand cost centres assigned from the unallocated list over the following days - the validation run at the end of each day measuring what remains. In the User Manual - Hierarchy and Cube Development, see the Hierarchies, Manage Levels, Manage Nodes and Validate sections.

6 Creating cubes

One hierarchy answers one question: how do these figures roll up through this structure? Management questions are usually two-dimensional: what did this department spend on that account family? A cube is Wevo's answer - the pairing of two validated hierarchies, conventionally an organisational structure and a chart of accounts, across which every intersection is summarised. Once a cube is built, any node from the first hierarchy can be crossed with any node from the second, at any level of either, and the figure is already computed and waiting.

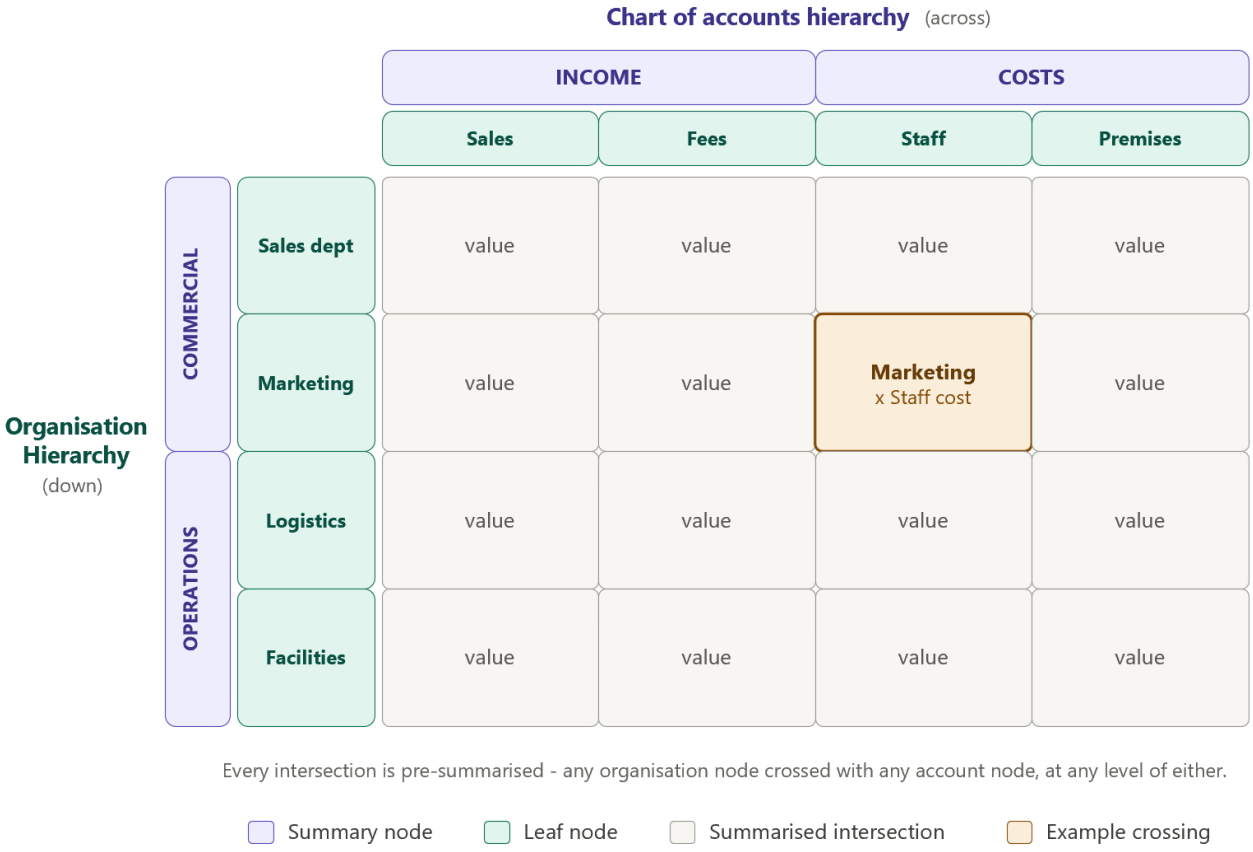

Figure 11 - A cube as the pairing of two hierarchies - every intersection of the organisation and account structures summarised in advance.

6.1 What defines a cube

A cube is defined by its two hierarchies together with a record type and a version - so actuals and budgets, or original and revised budget versions, are held as distinct, comparable slices. Against each cube the administrator activates the reporting years it should hold; Wevo carries up to nine years side by side, which is what allows the enquiry screens to show multi-year comparisons in a single view. Each hierarchy pairing also declares its drill-down start node - the node at which enquiry begins for users of that cube - and whether budget data is summarised alongside actuals for variance reporting.

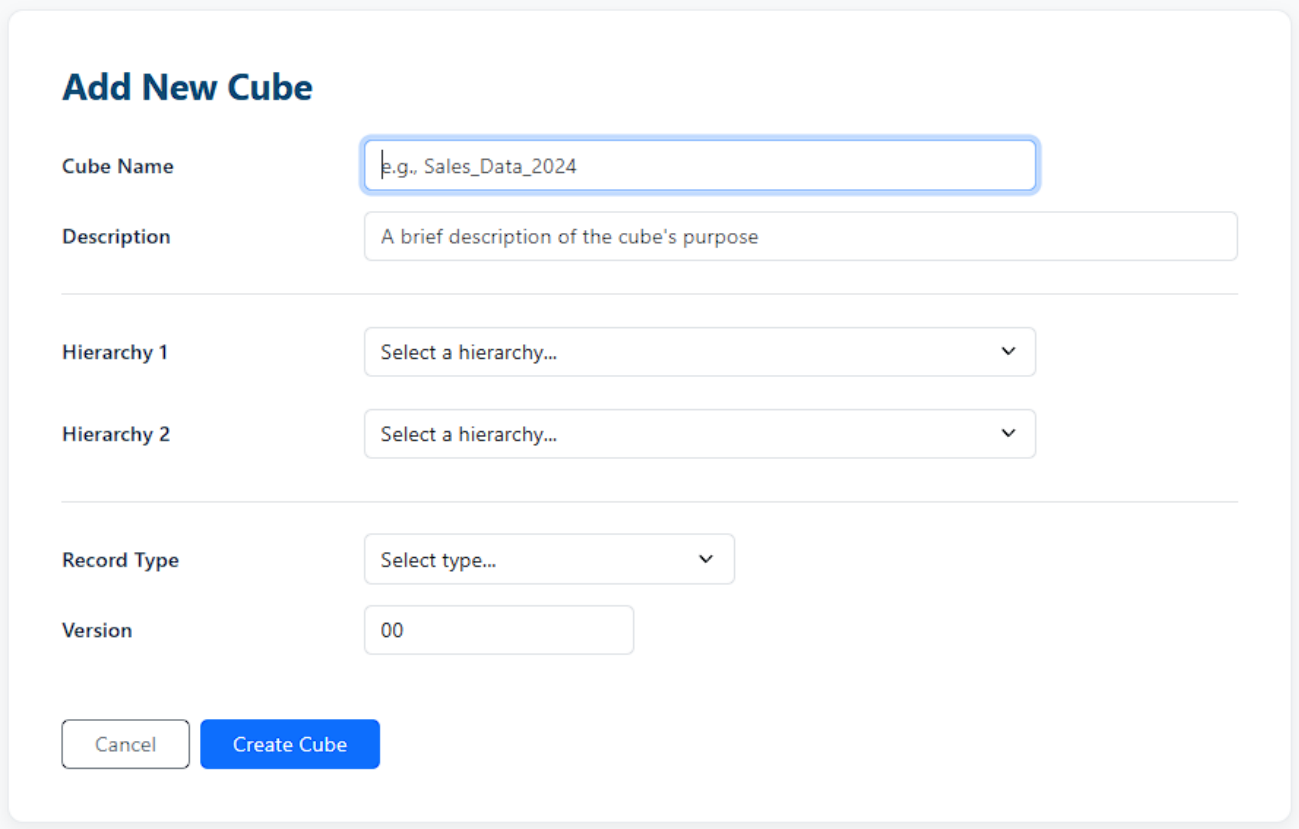

Figure 12 - Creating a cube - naming the pairing and choosing its two validated hierarchies.

Creation itself is a short form: a name, a description, a record type and version, and the two hierarchies. The cube detail view then manages everything about the pairing through its tabs: the summary years it holds, a copy facility for creating variants, and the build history that records every summarisation run against it.

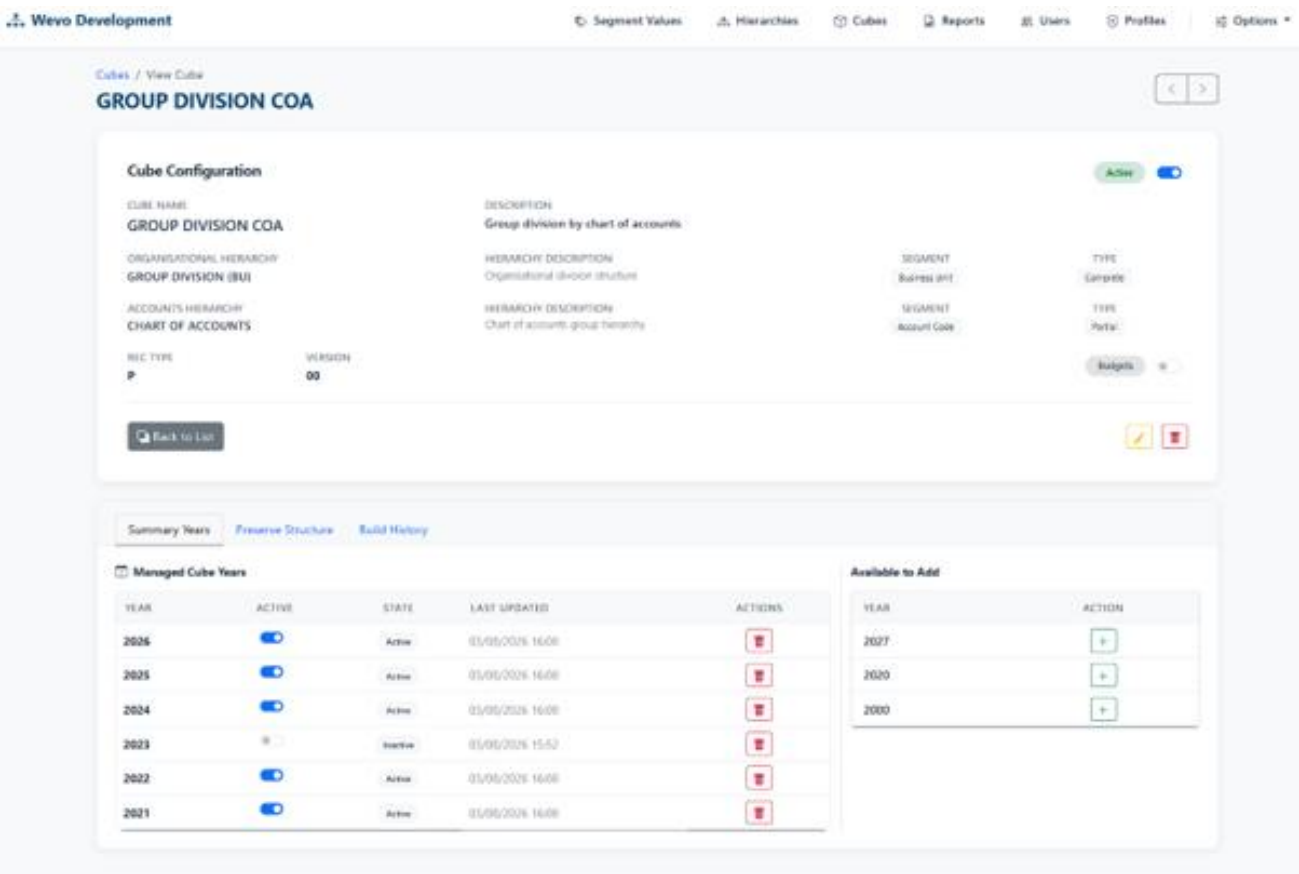

Figure 13 - The cube detail view - the pairing, its active years, and its build history.

6.2 The worked example - COST CENTRE COA

The banking group pairs GROUP DIVISION (CC) with CHART OF ACCOUNTS to create the cube COST CENTRE COA, record type P, version 00 - its primary actuals cube. Five reporting years are activated so that trend comparisons run back through prior years, and budget summarisation is switched on so every intersection carries actual against budget. A second cube, GROUP DIVISION COA, pairs the business unit hierarchy with the same chart of accounts, giving management the same figures through the legal entity lens - same data, different question, no duplication of effort.

In the User Manual - Hierarchy and Cube Development, see the Cubes section for the screens involved.

7 Feeding the cube - GL extraction

Structures without data are scaffolding. The final building block is the feed: the flow of figures from the GL into the cubes. Wevo's interface here is deliberately simple and deliberately arm's-length - the GL exports CSV files, the files land in a folder, and the Wevo summarisation engine does the rest. There is no direct connection to the GL database, no agent installed on the GL system, and nothing for the GL team to run except the extract they already know how to produce.

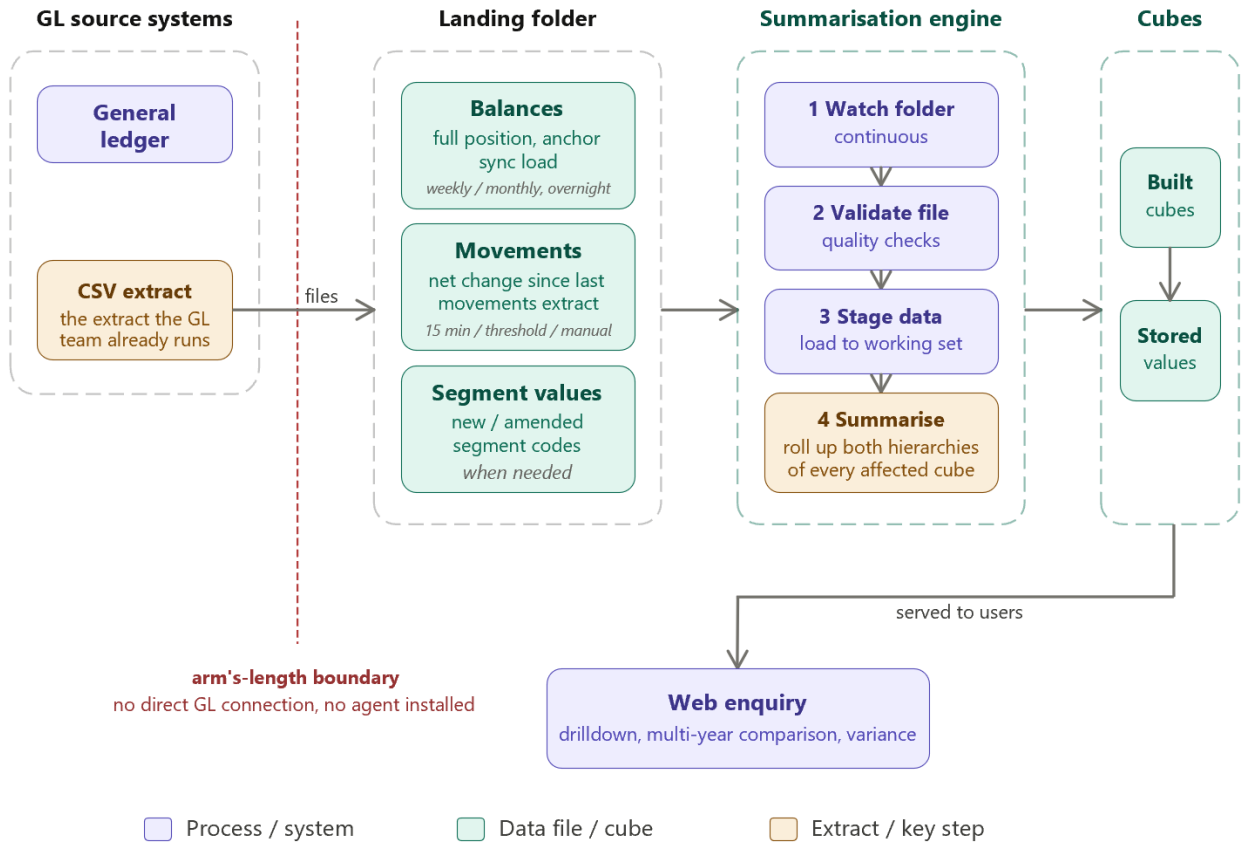

Figure 14 - End-to-end data flow - GL extract files, the summarisation engine, cube build, and web enquiry.

7.1 What flows, and when

Three kinds of financial file feed the cubes. Balances files carry the full GL position - the complete restatement of every code combination - and anchor the data, typically loaded on a weekly or monthly overnight build schedule. Movement files carry the GL transactions applied to the ledger since the last extract. Intelligence can be applied to this movements extract to keep the incremental update efficient: it can be triggered by elapsed time, by the total value of transactions posted, by a threshold number of transactions, or by a manual trigger - whichever comes first. Segment values files, carrying new or changed codes, complete the set. The engine watches its landing folder continuously, validates each arriving file, stages the data and runs the summarisation - every roll-up through both hierarchies of every affected active cube, computed and stored.

File type	Contains	Typical cadence
Balances	Full position for every code combination - the anchoring sync load	Weekly / Monthly overnight schedule

Movements	Net change (journal transactions) since the last GL movements extract	Every 15 minutes then trigger on exceeding a total amount of transaction values, a number of transactions threshold or a manual trigger
Segment values	New and amended segment value codes from each GL source system (full list)	Regular schedule (segment values overwritten each time) or when needed (changes to segment values)

7.2 Designing the extract strategy

The extract strategy is a business decision about data currency: how far behind the ledger may the reporting be? A daily balances load alone gives yesterday-close reporting, which suits many organisations. Adding intraday movements brings the cubes to within minutes of the GL, which transforms period-end: instead of waiting for the close pack, controllers watch the close happen. The strategy can differ by cube and can evolve - most adoptions begin with the daily load and add intraday movements once the value is visible.

The online reporting never stops for a load: each cube is rebuilt behind the scenes and released in a single switch, so users see either the old data or the new, never a half-built state, and cubes are released one by one as each completes.

7.3 Watching the feed

The Build Information page in the enquiry application is the business window onto the feed: every recent build, per cube and year, with its data type, build type, the source file that drove it and the number of rows updated. The status indicator on the navigation bar summarises the same story - what kind of data last arrived and when - so a user always knows how current the figures on screen are before quoting them.

Wevo GL Analytics Profile Hierarchies Cubes Build Information Status: Awaiting data; Movements applied 18/07 11:33

Build Information

Build Time	Cube Full Name	Year	Data Type	Build Type	Build Info	Update Count
18/07/2026 11:33:57	**COST CENTRE COA P 00**	2021	Movements	Incremental	small movements file 25/06/2026 ES	30
18/07/2026 11:33:57	**GROUP DIVISION COA P 00**	2021	Movements	Incremental	small movements file 25/06/2026 ES	30
18/07/2026 11:33:57	**GROUP DIVISION COA P 00**	2022	Movements	Incremental	small movements file 25/06/2026 ES	30
18/07/2026 11:33:56	**COST CENTRE COA P 00**	2023	Movements	Incremental	small movements file 25/06/2026 ES	30
18/07/2026 11:33:56	**COST CENTRE COA P 00**	2024	Movements	Incremental	small movements file 25/06/2026 ES	30
18/07/2026 11:33:56	**GROUP DIVISION COA P 00**	2024	Movements	Incremental	small movements file 25/06/2026 ES	30
18/07/2026 11:33:56	**COST CENTRE COA P 00**	2025	Movements	Incremental	small movements file 25/06/2026 ES	45
18/07/2026 11:33:56	**GROUP DIVISION COA P 00**	2025	Movements	Incremental	small movements file 25/06/2026 ES	45
18/07/2026 11:33:56	**COST CENTRE COA P 00**	2026	Movements	Incremental	small movements file 25/06/2026 ES	40
18/07/2026 11:33:56	**GROUP DIVISION COA P 00**	2026	Movements	Incremental	small movements file 25/06/2026 ES	40
18/07/2026 11:33:45	**COST CENTRE COA P 00**	2025	Movements	Incremental	smallmovements file 25/09/2025 NY	104,937
18/07/2026 11:33:32	**GROUP DIVISION COA P 00**	2025	Movements	Incremental	smallmovements file 25/09/2025 NY	282,296
18/07/2026 11:33:01	**COST CENTRE COA P 00**	2026	Movements	Incremental	smallmovements file 25/09/2025 NY	159,740
18/07/2026 11:32:43	**GROUP DIVISION COA P 00**	2026	Movements	Incremental	smallmovements file 25/09/2025 NY	890,434
18/07/2026 11:29:56	**COST CENTRE COA P 00**	2021	Balances	Full Build	balances 19/06/26 17:50	30
18/07/2026 11:29:56	**GROUP DIVISION COA P 00**	2021	Balances	Full Build	balances 19/06/26 17:50	30
18/07/2026 11:29:56	**GROUP DIVISION COA P 00**	2022	Balances	Full Build	balances 19/06/26 17:50	30
18/07/2026 11:29:56	**COST CENTRE COA P 00**	2023	Balances	Full Build	balances 19/06/26 17:50	30
18/07/2026 11:29:56	**COST CENTRE COA P 00**	2024	Balances	Full Build	balances 19/06/26 17:50	50
18/07/2026 11:29:55	**GROUP DIVISION COA P 00**	2024	Balances	Full Build	balances 19/06/26 17:50	64

1 2

Build Information

Build Time	Cube Full Name	Year	Data Type	Build Type	Build Info	Update Count
18/07/2026 11:28:56	**COST CENTRE COA P 00**	2025	Balances	Full Build	balances 19/06/26 17:50	5,104,536
18/07/2026 11:24:13	**GROUP DIVISION COA P 00**	2025	Balances	Full Build	balances 19/06/26 17:50	4,658,325
17/07/2026 19:48:20	**COST CENTRE COA P 00**	2026	Balances	Full Build	balances 19/06/26 17:50	5,103,254
17/07/2026 19:43:54	**GROUP DIVISION COA P 00**	2026	Balances	Full Build	balances 19/06/26 17:50	4,658,742

1 2

Figure 15 - Build Information - the business view of the feed: every build, its source file and its effect.

For the banking group, MasterGL is scheduled to export a balance file each Saturday evening for a full rebuild, movements files every thirty minutes after that, and a segment values file at six each morning. The first full load populates all five years of COST CENTRE COA in a single build; from then on, the group's reporting is minutes behind its ledger and stays there.

8 Using the reporting

Everything so far is preparation; this is the payoff. A business user opens the enquiry application, chooses a cube, sets their selection criteria, and is inside the figures. The criteria express, in accounting terms, exactly what the view should show: the data type (balances, movements or daily moves, with budgets alongside where held), the years to compare, the span of periods, the calculation basis (accumulated balance, net movement or period by period), the currency, the scaling, and the debit notation the reader prefers.

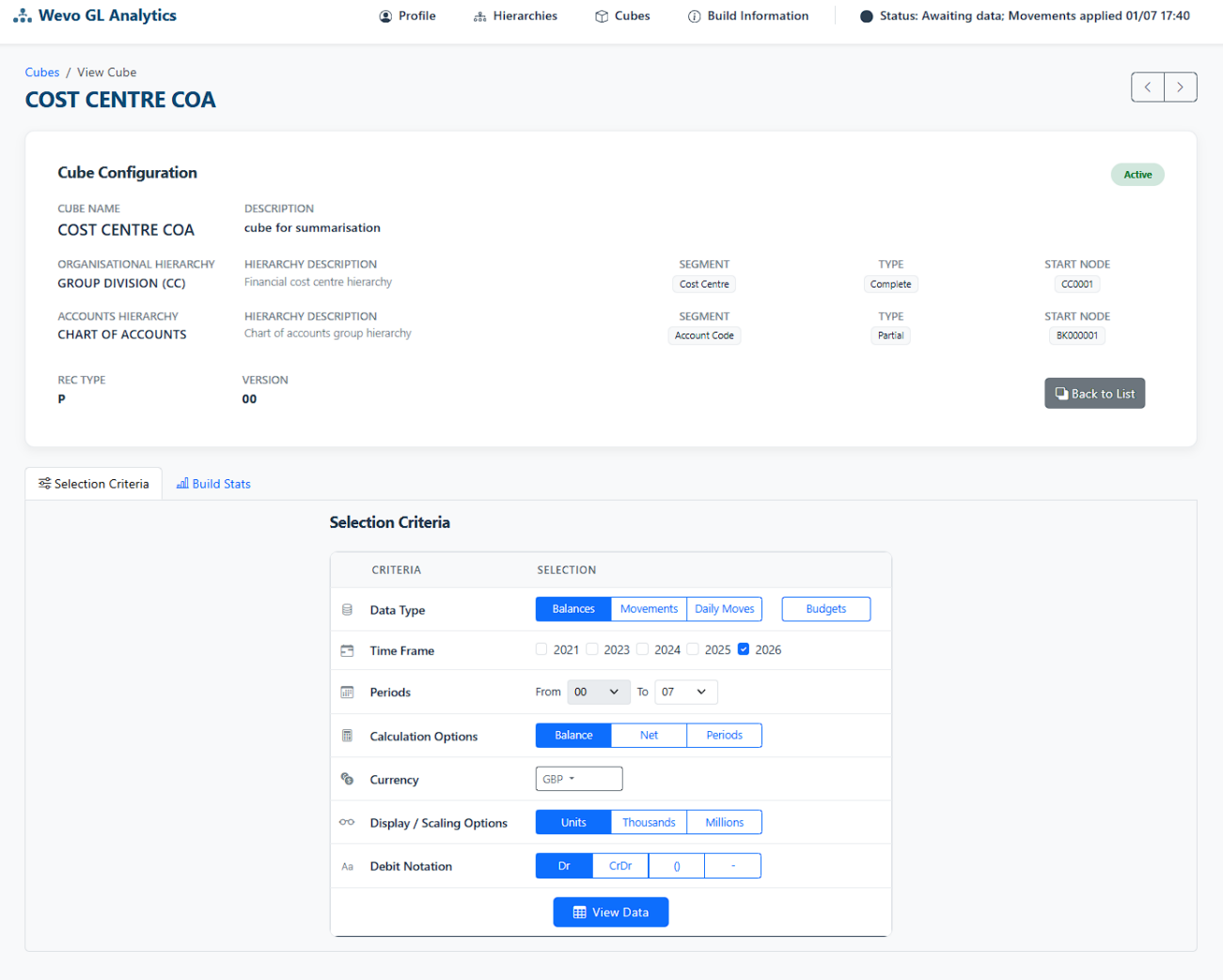

Figure 16 - Selection criteria - the accounting terms of the enquiry, set before the drill begins.

8.1 The drill-down

The drill-down transaction presents both hierarchies of the cube side by side - the organisational structure on the left, the accounts on the right - each showing the focused node in the context of its parents above and children below. Beneath them, the summary values table breaks the focused intersection down by the children of whichever side the user selects, with a total line that always reconciles to the figure above. Every navigation is a single click: drill into a child, step across siblings, pivot the breakdown from the organisation side to the accounts side, scroll the data columns across years and periods. Because every figure is pre-computed, each step lands in well under a second.

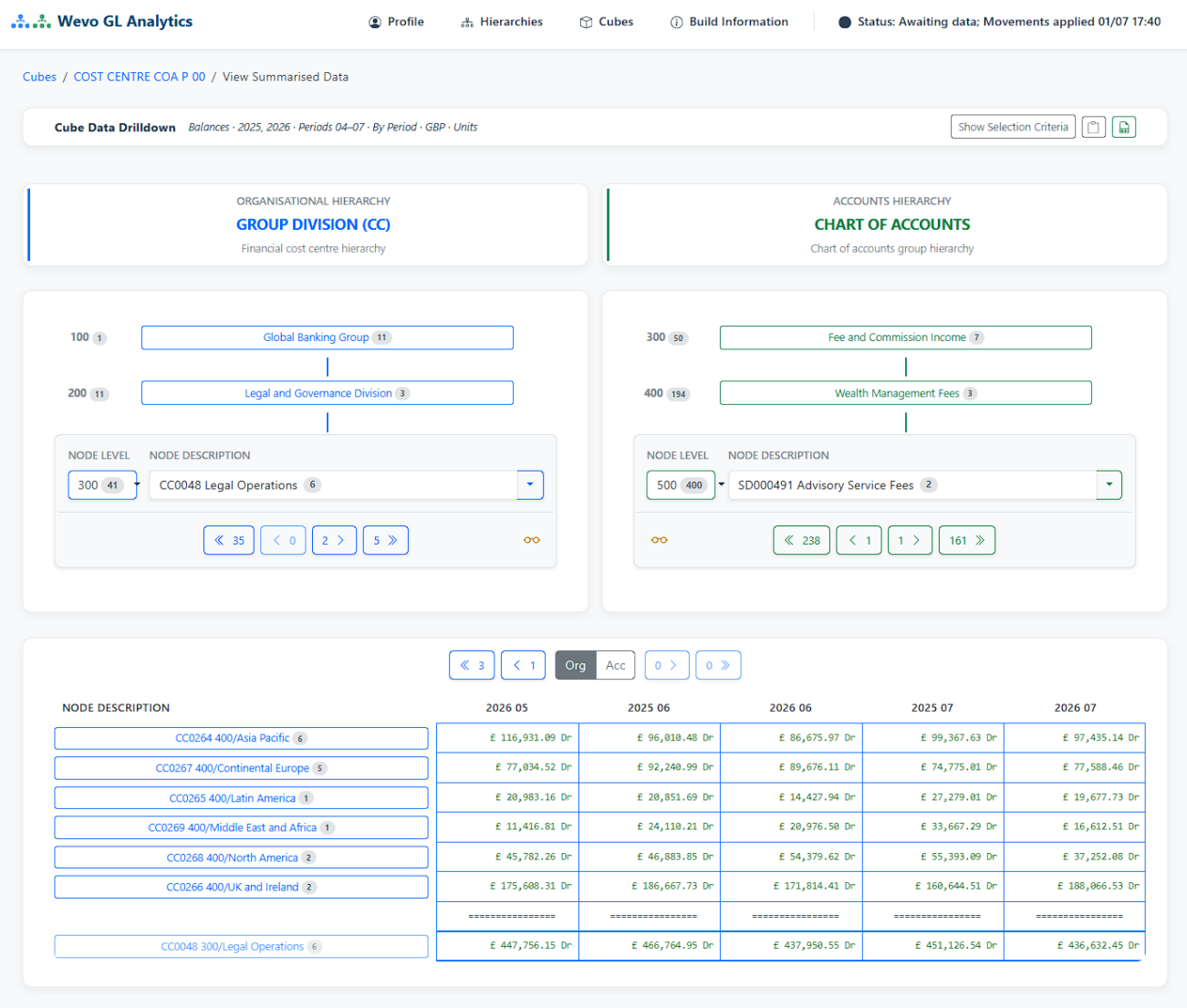

Figure 17 - The drill-down - both hierarchies in context, with the intersection broken down by the selected side.

For the banking group this is the moment the design proves itself: a controller starts at the Global Banking Group total for advisory service fees, drills through Corporate Banking to the Syndicated Lending Desk, pivots to the account side to see which fee accounts drive the figure, and exports the breakdown to a CSV file for the divisional pack - four clicks and an export, on figures thirty minutes old.

8.2 Beyond the drill

Three facilities round out daily use. Budgets and variance: where a cube summarises budget data, every intersection carries actual against budget at every level, with multiple budget versions held side by side. The watchlist: chosen organisation and account node pairs are monitored automatically as watched node pairs - on every incremental movements update in which a watched pair is recalculated, Wevo writes a time-stamped snapshot of its movements and balances, for each currency defined in the system parameters, building a longitudinal history for trend and anomaly review; watched pairs are flagged with a watchlist icon during ordinary drilling. Email reports: authorised users request formatted reports by email and receive them the same way, with no licence seat and no system access - the natural channel for the wider population who need figures but not a screen.

8.3 Who sees what

Access follows the hierarchy. Each user carries a profile, and each profile grants entry to chosen hierarchies at a chosen node - a divisional director's profile might open GROUP DIVISION (CC) at their division, so their consolidated view is their division and everything beneath it, and nothing beside it. Because security is expressed in the same structure as the reporting, it is administered in business terms and audited the same way. In the User Manual - Cube Drill-Down Enquiry, see the Cubes and Profile sections; user and profile administration is covered in the Hierarchy and Cube Development manual.

9 Operating and evolving

Adoption does not end at go-live; the structures must live with the business. Wevo's operating model is built around the certainty that organisations change, and its central instrument is the separation of development from production. Every hierarchy exists in two copies: the production copy that reporting reads, and the development copy where change is prepared. Restructures of any size are built and validated in development while production reporting continues untouched, and take effect only at promotion with a full balance build schedule.

9.1 Restructuring - as-is and as-was

The hard problem in management reporting is history. When departments move between divisions, should last year's figures be shown under the old structure or the new? Wevo's answer is both, by versioning hierarchies against years. Prior years read through the current structure for like-for-like comparison - the as-is view - while each year's structure as originally reported is retained and can be enquired on - the as-was view. Restructure freely; nothing is lost and nothing needs restating.

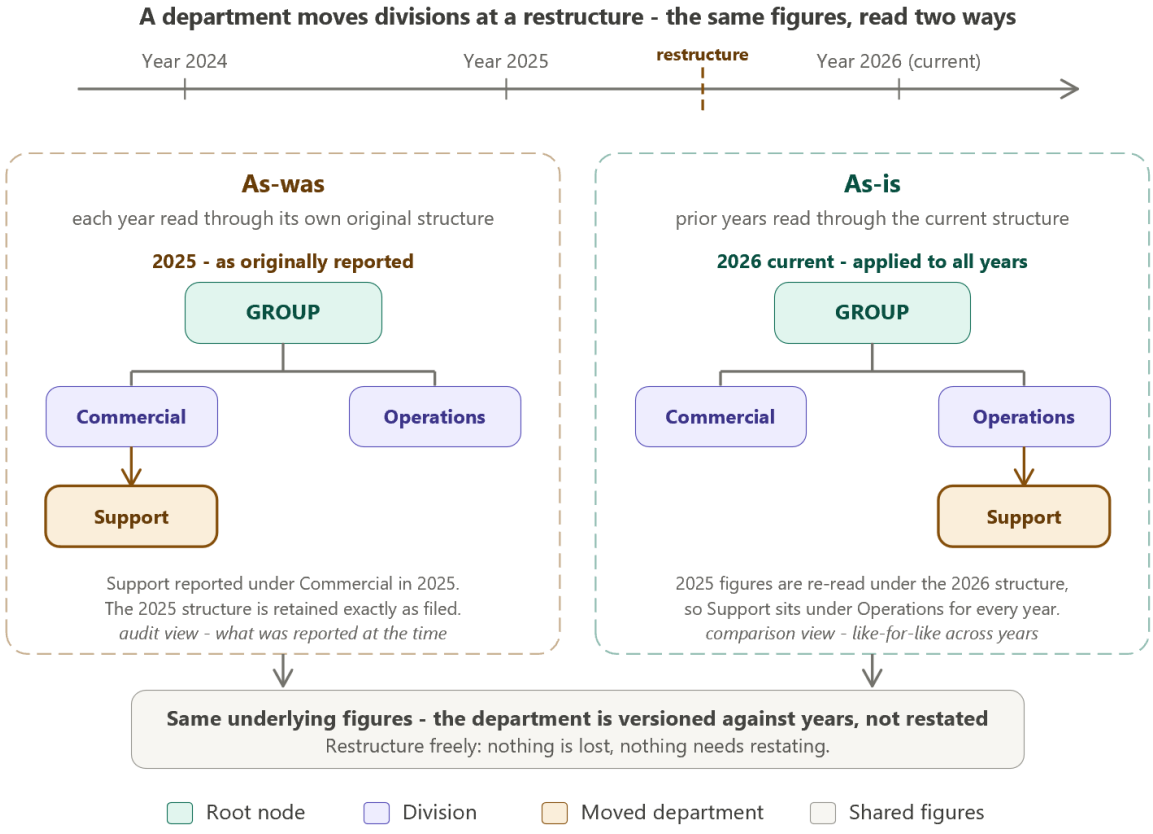

Figure 18 - As-is and as-was reporting - prior years read through the new structure, with each year's original structure retained.

9.1.1 Preserving a structure - the Preserve Structure function

A retained structure is created by the Preserve Structure function, a single operation on the cube detail view. The operator opens the cube, selects the Preserve Structure tab, supplies a name for the preserved cube - suffixed to identify what it preserves, for example GROUP 2025 - and a preserve description recording why it is being kept. One click then copies both hierarchies from

production, creates the cube on the copies, and leaves it dormant. The whole operation is a single transaction: it either completes or nothing is created.

The copy is taken from the production structure. Any hierarchy changes in progress in the development environment are separate and are not included - a preserved structure records what the business was actually reporting on, not what somebody happened to be building at the time. For this reason the function is available only on a development cube that has been summarised at least once; there must be a production reality to preserve.

The preserved cube is created inactive and holds no data at all. It takes no part in any build or incremental update and consumes no processing; for an enterprise group the whole structure occupies under 1 MB. It is a definition, not a dataset, and it sits there costing effectively nothing until somebody wants it.

When the historical view is wanted, reporting years are allocated to the preserved cube and the cube is set active. The data is built at that point from the retained segment-value figures, and the year can be interrogated through the structure that was actually in force. Years can be added or removed at will thereafter, and current years can also be allocated - to see how live data looks when read through a previous structure, the same as-was lens applied to figures the business is generating now.

Once created, a preserved structure cannot be amended, only deleted. Its hierarchies are locked and its composition is fixed; the only field that remains editable is the preserve description. This matters for audit. A structure that could be quietly adjusted invites the question of whether it is really what was in force at the time, whereas a structure that cannot be changed after creation answers that by construction. Deletion is straightforward when a preserved structure is no longer wanted, and removes both hierarchy copies with it, so the register does not accumulate orphans.

The result is two coexisting views over the same underlying figures: the current structure, through which prior years can be read like-for-like (the as-is view), and any number of preserved structures, through which any year - past or present - can be enquired on exactly as it was reported at the moment of preservation (the as-was view). Neither requires restatement, and the business chooses which to consult according to whether it wants comparability or historical fidelity.

9.1.2 Preserve Structure as a standing procedure

Because a preserved structure costs effectively nothing to hold, there is no judgement call to make about which structures might matter later, and no need to predict which reorganisation will be questioned in three years time. That is what makes a standing policy realistic rather than aspirational. This guide recommends the procedure below be adopted formally, with a named owner, rather than left to memory.

Owner. The hierarchy owner who signs off promotions is the natural owner of the preserve procedure, because the events that trigger it are the same events that drive a restructure. The financial control team holds the register of what has been preserved and why.

Triggers - preserve before the change, never after:

- Before any reorganisation - the classic case, and the one that cannot be recovered afterwards
- Before an acquisition, merger, or disposal, and again at completion
- At each year end, and monthly or quarterly as routine hygiene
- Before a chart of accounts change, an entity renumbering, or a cost centre rationalisation
- Before any change requested purely for management reporting - these are the ones nobody documents and everybody later needs

Recording. The preserve description is mandatory for exactly this reason: it is what makes the register readable years later. Record the event, not the date alone - "pre-reorganisation June 2026"

tells the next person what they are looking at; "June copy" does not. If the structure is not preserved before the change, reproducing those figures afterwards is extremely difficult and frequently impossible - and no amount of money fixes it at that point.

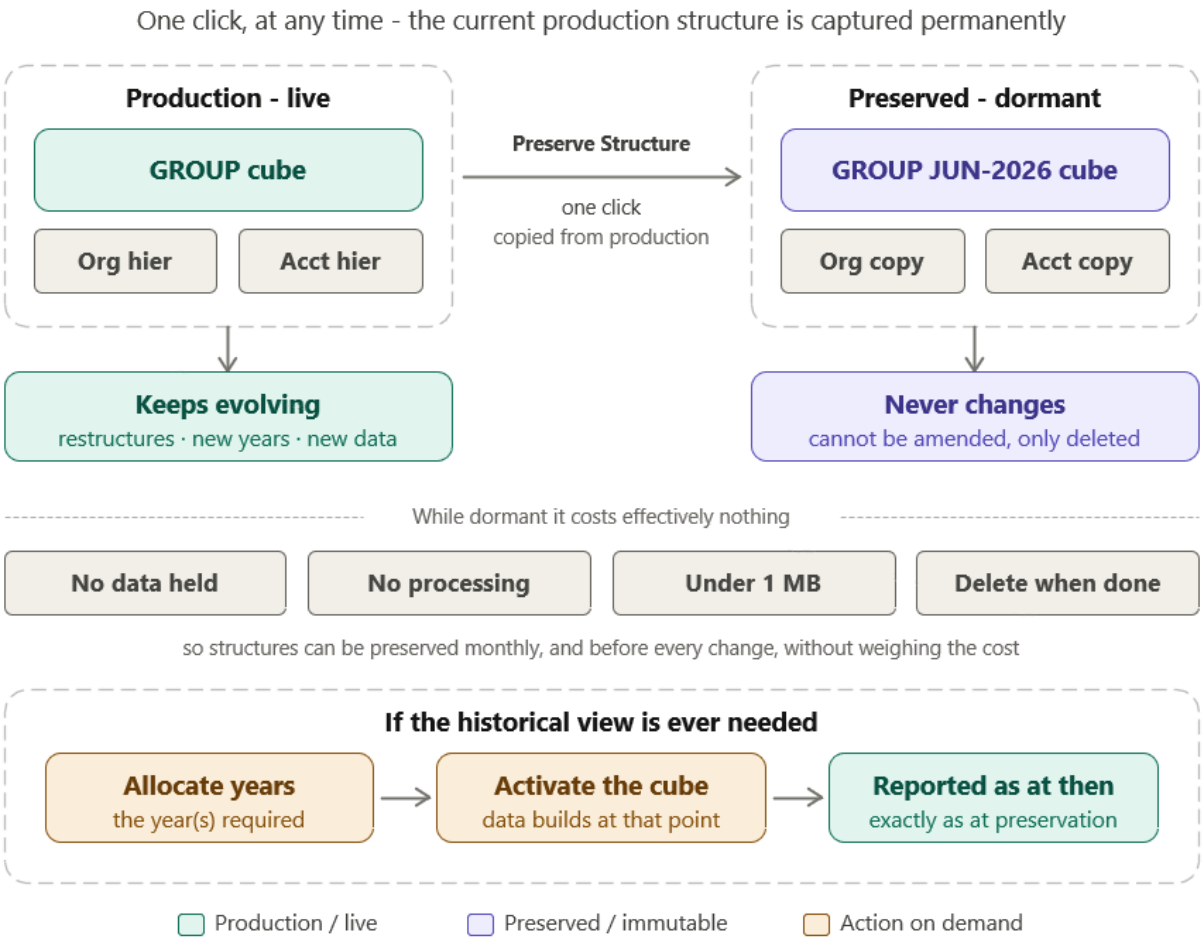

Figure 19 - A preserved structure alongside the live one - the preserved copy holds the structure of the moment while the current structure evolves, with reporting years allocated against the preserved cube on demand.

9.2 The close, and the year end

Wevo carries sixteen periods against each year - either twelve trading periods plus four close periods, or thirteen plus three - so year-end and audit adjustments flow through every hierarchy exactly as ordinary postings do. The optional year-end read-only archive preserves a complete, frozen instance of the reporting as at the close, satisfying audit without constraining the live system. During sensitive windows an administrator can lock hierarchies against change, and the development instance gives the team somewhere to rehearse the next year's restructures while the close proceeds undisturbed.

9.3 The operating rhythm

In steady state the human effort is modest and belongs to the business, not to IT. Daily: glance at Build Information and the status indicator - the feed is otherwise autonomous. Monthly: assign newly arrived segment values from the unallocated list, review watchlist snapshots, and action any joiner and leaver changes in users and profiles. Monthly also: preserve the current structure as

routine hygiene - one click, under 1 MB, and it removes the risk entirely. Quarterly, or as reorganisations require: preserve the structure first, then prepare restructures in development, validate, obtain the hierarchy owner's sign-off, and promote. The worked example settles into exactly this rhythm, with the financial control team owning the structures outright and the GL team's involvement ending at the nightly extract.

10 The adoption roadmap

The chapters of this guide are, in order, the phases of an adoption. The table below draws them together into a working roadmap: what each phase produces, who leads it, and where the mechanics are documented. Timescales depend on the size of the code population and the state of the design going in; the banking group of our worked example ran the sequence in six weeks, with hierarchy design the longest phase - as it should be, because design is where the business value is decided.

Phase	Produces	Led by	Screens involved
1. Review segment values	Confirmed code populations, owners, corrected descriptions	Finance data owner	Segment Values
2. Design hierarchies	Agreed structures, levels, naming convention, sign-off	Hierarchy owners	Design workshops (off-system)
3. Build and validate	Validated hierarchies, empty unallocated lists	Adoption team	Manage Levels, Manage Nodes, Validate
4. Create cubes	Named cubes with active years and budget settings	Adoption team	Cubes, Cube detail
5. Establish the feed	Scheduled extracts, first full load, verified builds	GL team and adoption team	Build Information
6. Set up access	Profiles, users, report assignments	Finance administrator	Users, Profiles, Reports
7. Promote and go live	Production hierarchies, live enquiry	Hierarchy owners	Hierarchy detail, Drill-Down
8. Operate and evolve	The steady-state rhythm of chapter 9	Financial control	All of the above

Appendix - Glossary

Term	Meaning
Segment	One field of the GL accounting key - for example Business Unit, Cost Centre or Account Code.
Segment value	An individual code within a segment, defined and owned by the source GL.
Hierarchy	A tree of named nodes, built on one segment, through which figures are aggregated.
Level	A named tier of a hierarchy. Every node belongs to exactly one level.
Root node	The single node at the top of a hierarchy - the consolidated view.
Leaf node	A bottom-level node representing one real GL segment value, named identically to it.
Summary node	A Wevo-defined grouping above the leaves, existing only for reporting.
Complete hierarchy	A hierarchy in which every segment value has a place; reconciles to the GL total.
Partial hierarchy	A hierarchy covering a chosen subset of segment values, for a focused view.
Validation	The rule set a hierarchy must pass - structure, links and coverage - before promotion.
Promote	The action that releases a validated development hierarchy to production reporting.
Cube	The pairing of two validated hierarchies, with a record type and version, across which every intersection is summarised.
Record type	The category of GL data - actuals, budgets, forecasts - held as distinct slices.
Version	A numeric identifier distinguishing variants within a record type, such as budget revisions.
Balances / Movements	Full position, and net change accumulated since the last cleardown - the two main feed types.
Summarisation engine	The batch component that receives extract files and computes every roll-up in advance.
Drill-Down	The enquiry that navigates both hierarchies of a cube from any figure to its detail.
Watchlist	Chosen organisation and account node pairs monitored with a time-stamped snapshot on every incremental movements update the pair is recalculated.
As-is / As-was	Prior years read through the current structure, and each year's original structure retained for history.

Preserve Structure	The one-click operation that copies a cube's two hierarchies from production and creates a dormant cube on the copies, preserving the structure of that moment permanently. The result cannot be amended, only deleted.
Preserved structure	The artefact created by Preserve Structure - a dormant cube and its two locked hierarchy copies. Holds no data, occupies under 1 MB, and generates figures only when years are allocated and the cube is activated.
Profile	The security definition granting a user entry to chosen hierarchies at chosen nodes.