
MAKING EVERY WILLOW AREA CONTIGUOUS: OBSERVATIONS ON STORAGE AND RANGE RECONCILIATION

Elvio G. Amparore

Dipartimento di Informatica, Università di Torino
Torino, Italy
elviogilberto.amparore@unito.it

October 2, 2026

ABSTRACT

Willow¹ is a data model for synchronisable hierarchical peer-to-peer data, while Earthstar² is a distributed storage protocol and implementation project whose current v11 development uses Willow. Within a Willow namespace, Entries are described by three coordinates: subspace, hierarchical path, and timestamp. The specifications define both generic 3dRanges and the more restricted notion of an *Area*. Areas are not arbitrary three-dimensional boxes: the subspace is either one exact value or unrestricted, the path denotes a prefix cone, and only the timestamp is an ordinary interval.

Two concrete implementations are analyzed as reference points: a TypeScript implementation maintaining three lexicographic coordinate orders over the Entry set, and a Rust implementation maintaining one subspace–path–timestamp order and filtering larger bounding scans when queries do not align with it. Both store one representative per logical Entry in each maintained coordinate index. In this design family, no lexicographic permutation of subspace, path, and timestamp makes every Willow Area contiguous, and even maintaining all six permutations leaves some Area shapes non-contiguous.

This paper describes a different approach, called *AreaScopeIndex*. It denormalizes the derived membership relation between Entries and *Area scopes*: for every Entry, membership is materialized for every ancestor path prefix, both in the Entry’s exact subspace and in an unrestricted-subspace scope. Entries in each materialized scope are stored in one timestamp-first canonical order. An Area then becomes one materialized scope plus one timestamp interval, so **every Willow Area, including the path-plus-time and subspace-plus-path-plus-time cases, is exactly one contiguous range of one ordered index**. On a range-summarizable B+-tree of height h , locating or summarizing any Area costs $O(h)$, enumerating m matching Entries costs $O(h + m)$, and rank/select operations cost $O(h)$, independently of Area shape. This uniform read-side access is obtained by bounded write-time denormalization: an Entry at depth d contributes exactly $2(d + 1)$ index representatives, and under an effective maximum depth D its maintenance costs $O(Dh)$. The contiguity result itself is independent of path depth.

The linearization can be used most directly by a modest change to Willow range-based set reconciliation. Willow RBSR is currently expressed over generic 3dRanges without assuming a common efficient order. Once an Area is represented as one ordered range, balanced refinement can instead operate on *slices*: an Area scope together with an interval of the canonical within-scope order. Slices contain Areas, are closed under balanced rank-based splitting and intersection, and remain single *AreaScopeIndex* ranges throughout reconciliation. A slice-based RBSR variant requires peers to agree on the same canonical order κ and to exchange slice descriptors; under these explicit protocol assumptions, a range-summarizable B+-tree provides $O(h)$ summaries and rank/select operations and $O(bh)$ balanced b -way cuts.

¹<https://willowprotocol.org/>

²<https://earthstar-project.org/>

Keywords Willow · Earthstar · hierarchical indexing · B+-trees · range queries · 3dRange · range-based set reconciliation

1 Introduction

Willow is a data model and associated family of synchronization specifications for hierarchical peer-to-peer data stores [15]. Earthstar is a distributed storage protocol and implementation project whose current v11 development uses Willow as its data model [4]. Willow assigns structured hierarchical paths to byte payloads, separates Entries into subspaces, associates timestamps with assignments, and groups independent collections into namespaces. The resulting model provides substantially more structure than a flat key–value set while remaining regular enough to support replication and range-based synchronization.

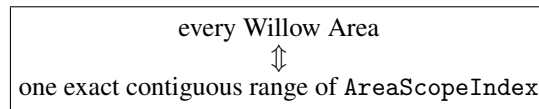
Within one namespace, a Willow Entry can be viewed for storage purposes as a point with three relevant coordinates: a subspace identifier, a hierarchical path, and a timestamp. Willow then defines two range languages over these coordinates. The application-facing language is the *Area*: one exact subspace or all subspaces, one path prefix and all of its extensions, and one contiguous time interval [16]. The more general language is the *3dRange*: a Cartesian product of independent intervals over subspaces, paths, and timestamps [17]. Every *Area* can be expressed as a *3dRange*, but the converse does not hold.

The distinction is more than terminological. An *Area* does not contain three arbitrary interval predicates. Its subspace predicate is exact-or-any; its path predicate is hierarchical prefix membership; only the timestamp predicate is an ordinary ordered interval. Treating all three coordinates as generic dimensions therefore exposes more geometric freedom than the query language actually needs.

The Willow storage notes treat storage explicitly as a three-dimensional indexing problem: they discuss multidimensional trees, Z-order linearization, and a simpler design based on multiple lexicographic coordinate orders [14]. The *willow-js* roadmap likewise lists a Z-ordered skiplist as a prospective *Storage3d* implementation [5]. The material examined here does not describe denormalizing *Area*-scope membership as an indexing strategy; *AreaScopeIndex* therefore explores a different representational direction rather than another multidimensional linearization. Willow’s 3d range-based reconciliation specification does not prescribe a linearization and explicitly requires a commutative fingerprint combination for that reason [17].

Under this model, the limitation of coordinate-permutation indexes is real but narrower than the Willow data model itself: any *TripleStore*-like design that keeps one representative per logical Entry in each lexicographic coordinate index necessarily leaves some *Area* shapes non-contiguous, even if all coordinate permutations are maintained.

The representation can instead be changed rather than the coordinate permutation. *AreaScopeIndex* is a denormalized index over a derived relation: whether an Entry belongs to a given non-temporal *Area* scope. For every Entry, one index representative is created for every ancestor path prefix, both under its exact subspace and under an unrestricted-subspace scope. The timestamp interval is not materialized. Instead, all representatives of one scope are stored in a single timestamp-first canonical order. An *Area* query is therefore reduced to equality on one materialized scope plus one interval on the next key coordinate. This yields the central result:



This is a representational result and does not depend on small path depth. A path-depth bound is needed only to bound write and space amplification.

The one-dimensional representation also suggests a modest change to reconciliation. Willow RBSR starts from *Areas* but recursively describes mismatching subsets using generic *3dRanges*. Balanced partitioning of an *Area* by Entry count does not, in general, produce another *Area*; with equal timestamps it may not even produce one box. The current formulation does not assume a shared efficient linearization. Once an *Area* is already represented by one canonical ordered range, however, the recursive range language can remain in that order. Such a range is called a *slice*: an *Area* scope together with an interval of the canonical within-scope order.

The argument proceeds in five steps:

1. Willow *Areas* are a restricted family of three-dimensional boxes, not generic 3D queries.
2. *TripleStore*-like lexicographic indexes that keep one representative per logical Entry in each maintained coordinate order cannot make every *Area* contiguous.

3. `AreaScopeIndex` changes the representation by materializing Area-scope membership, reducing every Area query to equality on one scope plus one timestamp interval.
4. Once this order exists, generic `3dRanges` reintroduce multidimensional structure into recursive RBSR refinement that is not needed by the local storage representation.
5. Slices preserve the canonical one-dimensional structure throughout balanced reconciliation and are therefore a more natural range language once the store has been reduced to one-dimensional scope ranges; `AreaScopeIndex` serves them with the same efficient ordered-range primitives.

Section 3 fixes the two implementation snapshots used as reference points and names them `TripleStore` and `PersistentStore`. All software observations are tied to those explicit versions and inspection dates rather than intended as claims about every possible Willow backend.

2 Willow Data Model and Range Geometry

This section states the parts of Willow needed by the indexing and reconciliation discussion, following the terminology of the Willow specifications [15, 16, 17]. A Willow *Area* is the standard grouping used below: it selects one exact subspace or all subspaces, one path prefix and all of its extensions, and one time interval. The constituent concepts are introduced first and the Area semantics are formalized in Section 2.3.

2.1 Namespaces and subspaces

A Willow *namespace* identifies an independent collection of Entries under one `NamespaceId` [15]. Subspaces, paths, timestamps, and the overwrite/pruning rules are interpreted within that namespace. The analysis fixes one namespace at a time; a physical implementation may use one tree or keyspace per namespace, or prepend the `NamespaceId` to all index keys without changing any result below.

Let $\mathcal{S}$ be the totally ordered set of subspace identifiers in the fixed namespace. Every Entry e has

$$s(e) \in \mathcal{S}.$$

Subspaces are semantically meaningful in Willow and interact with write authorization [15]. In an Area, however, the subspace component has only two modes: select one exact identifier, or select all subspaces.

Let

$$\text{ANY} \notin \mathcal{S}$$

be a distinguished symbol representing the unrestricted mode, and define

$$\mathcal{S}_\alpha = \begin{cases} \{\alpha\}, & \alpha \in \mathcal{S}, \\ \mathcal{S}, & \alpha = \text{ANY}. \end{cases}$$

Thus an Area never requests an arbitrary subspace interval.

2.2 Paths as prefix cones

A Willow Path is a finite sequence of Components. Write

$$p = (c_1, c_2, \dots, c_d),$$

and let $|p| = d$ denote its depth. The empty path is denoted by ϵ . Willow bounds component count and total path length through data-model parameters [15, 13].

Prefix is defined component-wise. For paths p and q , write

$$p \preceq q$$

when the first $|p|$ components of q are exactly the components of p . Consequently, `/room/A` is a prefix of `/room/A/x`, but not of `/room/A1`.

For indexing, a path is best viewed as the apex of a *prefix cone*.

Definition 1 (Path cone). For $p \in \mathcal{P}$,

$$\mathcal{C}(p) = \{q \in \mathcal{P} \mid p \preceq q\}.$$

For example,

$$\mathcal{C}(/room/A) = \{/room/A, /room/A/x, /room/A/photos, \dots\},$$

while $/room/A1$ is excluded. The empty path gives the universal cone,

$$\mathcal{C}(\epsilon) = \mathcal{P}.$$

For a path $q = (c_1, \dots, c_d)$, define

$$\text{Pref}(q) = \{\epsilon, (c_1), (c_1, c_2), \dots, (c_1, \dots, c_d)\}.$$

Then

$$|\text{Pref}(q)| = d + 1$$

and, crucially,

$$p \preceq q \iff p \in \text{Pref}(q). \quad (1)$$

This equivalence is the hierarchical relation materialized by `AreaScopeIndex`.

Willow paths are totally ordered lexicographically by components, with a proper prefix preceding its extensions. Under this order, prefix cones are intervals.

Lemma 1 (Cones are path intervals). *For every path p , $\mathcal{C}(p)$ is a convex interval of the Willow path order. Write it as*

$$\mathcal{C}(p) = [p, p^+),$$

where p^+ is the least path greater than every extension of p , or an open-end sentinel if no such finite path exists.

Proof. The least element of $\mathcal{C}(p)$ is p . Let $a, c \in \mathcal{C}(p)$ and $a < b < c$. If $p \not\preceq b$, either b is a proper prefix of p , which implies $b < p \leq a$, or b differs from p at a first component position. Since a and c agree with p through all components of p , that differing component would force either $b < a$ or $b > c$. Both contradict $a < b < c$. Hence $b \in \mathcal{C}(p)$. $\square$

2.3 Time and Areas

Let $\mathcal{T}$ be the totally ordered timestamp domain. Willow timestamps are unsigned 64-bit integers, and Willow ranges are half-open [15, 16]. Write

$$I = [t_0, t_1)$$

for a finite interval and $[t_0, \infty)$ for an open-ended one.

Ignoring payload and authorization metadata that does not affect membership, represent an Entry as

$$e = (s(e), p(e), t(e)).$$

An Area is represented as

$$A = (\alpha, p, I),$$

where $\alpha \in \mathcal{S} \cup \{\text{ANY}\}$, $p \in \mathcal{P}$, and I is one time interval.

Definition 2 (Area semantics). *The Entries in $A = (\alpha, p, I)$ are*

$$\mathcal{A}(\alpha, p, I) = \{e \mid s(e) \in \mathcal{S}_\alpha \wedge p(e) \in \mathcal{C}(p) \wedge t(e) \in I\}. \quad (2)$$

Equivalently,

$$\mathcal{A}(\alpha, p, I) = \mathcal{S}_\alpha \times \mathcal{C}(p) \times I.$$

Definition 3 (Area scope). *The scope of an Area is its non-temporal selector*

$$\gamma = (\alpha, p),$$

which denotes $\mathcal{S}_\alpha \times \mathcal{C}(p)$. An Area can therefore be written equivalently as (γ, I) : one Area scope plus one time interval.

The three predicates are therefore asymmetric:

Dimension	Area predicate
Subspace	one exact value or all subspaces
Path	one prefix cone
Time	one ordered interval

Table 1: The eight canonical Willow Area query shapes.

Shape	Subspace	Path	Time
FULL	any	all	all
T	any	all	range
S	exact	all	all
P	any	prefix	all
S+P	exact	prefix	all
S+T	exact	all	range
P+T	any	prefix	range
S+P+T	exact	prefix	range

2.4 Generic 3dRanges

Willow also uses a more general Cartesian range language. A 3dRange can be modelled as

$$R = J_S \times J_P \times J_T,$$

where J_S , J_P , and J_T are arbitrary half-open intervals of the subspace, path, and timestamp orders, respectively [17].

By Lemma 1, every Area is a 3dRange:

$$\mathcal{A}(\alpha, p, I) = \begin{cases} \mathcal{S} \times [p, p^+) \times I, & \alpha = \text{ANY}, \\ \{\alpha\} \times [p, p^+) \times I, & \alpha \in \mathcal{S}. \end{cases}$$

The inclusion is strict. A 3dRange can select, for example, a proper interval of subspaces or a path interval that is not a prefix cone. Such ranges have no direct Area equivalent.

This strict inclusion is important later:

$$\boxed{\text{Areas} \subsetneq \text{3dRanges.}} \quad (3)$$

The application query language is therefore narrower than the recursive range language currently used for Willow RBSR.

2.5 Eight canonical Area shapes

Each Area coordinate can be effectively restrictive or universal. The subspace is either ANY or exact, the path is either ϵ or a proper prefix, and the time interval is either the full domain or a proper range. This yields the eight canonical shapes in Table 1.

This classification will be used throughout the paper to distinguish storage costs that depend on query shape from those that do not.

3 Two Reference Implementations

This section examines how two concrete Willow implementations map Area queries to ordered storage. They are used as reference points for the structural analysis that follows, not as an exhaustive survey of possible backends.

Throughout the remainder of the paper, `TripleStore` denotes the `TripleStore` implementation in the `earthstar-project/willow-js` repository, published as `@earthstar/willow` version 0.6.1, with the main-branch source and documentation inspected on 2 October 2026 [5]. `PersistentStore` denotes `willow25::PersistentStore` in `crate willow25` version 0.7.9, published on 27 August 2026, with its source inspected on 2 October 2026 [18]. These names are used without repeating repository or version information below.

3.1 TripleStore

`TripleStore` is the extant `Storage3d` implementation identified by the inspected `willow-js` documentation. It maintains three differently ordered `SummarisableStorage` instances [5]. The corresponding Willow storage notes discuss the three one-dimensional orders

$$SPT = (\text{subspace}, \text{path}, \text{time}),$$

$$PTS = (path, time, subspace),$$

and

$$TSP = (time, subspace, path)$$

as a simple alternative to explicitly multidimensional structures [14].

Prefix relationships are handled separately through a `PrefixIterator` abstraction, with a radix-tree implementation listed in the repository [5]. Thus each logical Entry participates once in each maintained full-data ordering, but its ancestor path prefixes are not materialized as additional representatives in those summarizable indexes.

The three orders cover several Area shapes exactly. T aligns with TSP, S with SPT, P with PTS, and S+P with SPT. Other combinations are not one exact interval of any maintained order. They require either multiple fragments or a larger interval followed by filtering. In the worst case, both the number of exact fragments and the amount of overscan can grow linearly with the database size.

The same repository lists a Z-ordered skiplist as a desired future `Storage3d` implementation [5]. Together with the storage notes, this is consistent with treating the problem primarily as one of indexing or linearizing generic three-dimensional coordinates.

3.2 PersistentStore

`PersistentStore` uses `Fjall` for its Entry index and maintains one SPT-ordered `Fjall` key per logical Entry in the namespace keyspace; payload data are stored separately. The source-level helper `encode_spt_key` explicitly encodes

$$(subspace, path, timestamp)$$

in that order [18].

For `get_area`, the Area is converted to a generic three-dimensional range. Lower and upper SPT corner keys are constructed, the corresponding `Fjall` interval is scanned, and each candidate is decoded and tested again with the generic range-membership predicate before being emitted [18].

Let m_A be the number of Entries actually belonging to Area A , and let B_A be the number of stored keys in the scanned SPT bounding interval. Then

$$m_A \leq B_A \leq N.$$

The physical work depends on B_A , not only on m_A . For poorly aligned shapes, B_A can be $\Theta(N)$ even when m_A is small.

The asymmetry is particularly explicit in `forget_area`. With one exact subspace, the implementation can begin from a subspace/path prefix. With any subspace, the inspected source starts from an iterator over the whole namespace and filters candidates [18].

3.3 Structural comparison

The two implementations make different engineering choices, but both preserve the original coordinates and keep one representative for each logical Entry in each maintained coordinate index. Table 2 summarizes the resulting access shapes and compares them with `AreaScopeIndex`, defined formally in Section 5. The table is structural rather than a microbenchmark. “Fragments/overscan” means that no maintained order makes the result one exact interval in the general case; the stated worst case is for index work beyond the unavoidable output of matching Entries. N_s denotes the number of Entries in a selected subspace.

Table 2 concerns query/enumeration access. Reconciliation also imposes an additional summarization requirement. In the inspected `TripleStore` implementation, `summarise` uses the `TSP SummarisableStorage` for the requested three-dimensional range rather than selecting among the three orders as an ordinary query can; its reconciliation cost therefore follows that maintained order. `PersistentStore`, by contrast, exposes Area retrieval through its Store API but no maintained range-fingerprint or balanced range-split primitive in the analyzed snapshot [5, 18]. `AreaScopeIndex` uses the same single interval for both retrieval and cached B+-tree summarization.

The next section makes precise why adding further coordinate permutations cannot remove the query-shape asymmetry while retaining one representative per Entry per index.

4 Limits of Coordinate-Permutation Indexes

The analyzed implementations use a common coordinate-index pattern: each maintained index stores one representative for every logical Entry and orders those representatives lexicographically by the original Willow coordinates. `TripleStore`

Table 2: Area access by query shape. TripleStore and PersistentStore are the implementation snapshots fixed above. Enumeration of m returned Entries always adds the unavoidable $O(m)$ output cost.

Shape	TripleStore	PersistentStore	AreaScopeIndex
FULL	one exact range	full SPT range	one exact range
T	one exact TSP range	bounding scan; worst $\Theta(N)$	one exact range
S	one exact SPT range	SPT-aligned	one exact range
P	one exact PTS range	may span all subspaces; worst $\Theta(N)$	one exact range
S+P	one exact SPT range	SPT-aligned for the cone	one exact range
S+T	fragments/overscan; worst $\Theta(N_s)$	overscan inside s ; worst $\Theta(N_s)$	one exact range
P+T	fragments/overscan; worst $\Theta(N)$	bounding scan; worst $\Theta(N)$	one exact range
S+P+T	fragments/overscan; worst $\Theta(N)$	prefix-local overscan; worst linear in selected cone	one exact range

maintains three such orders, while PersistentStore maintains one. The following result applies to any *TripleStore*-like extension of this design, including a hypothetical implementation that maintains all six permutations of S , P , and T .

Formally, let an index order Entries lexicographically by some permutation π of the three original axes S , P , and T , with exactly one representative per logical Entry in that index. Multiple such indexes may be maintained simultaneously.

A query shape is called π -contiguous if, for every possible store and every Area of that shape, the matching Entries form one interval under π .

For the following proposition, the subspace constraint is a singleton when present, while path and time constraints are assumed nontrivial: the path cone contains at least two possible paths and the time interval at least two timestamps.

Proposition 1 (Lexicographic contiguity criterion). *Let X be the set of constrained axes of an Area shape. The shape is π -contiguous if and only if the axes in X form a prefix of π and every constrained axis in that prefix except possibly the last is restricted to a singleton.*

Proof. If the constrained axes form a prefix of π and every constrained axis except possibly the last is fixed to one value, the Area fixes the leading coordinates, restricts the next coordinate to one interval, and leaves all remaining coordinates unrestricted. Such a set is one lexicographic interval.

Conversely, suppose an unconstrained axis precedes a constrained one. Three coexisting Entries can be chosen with increasing values on the earlier unconstrained axis, while the later constrained coordinate is inside the Area for the first and third Entries and outside it for the middle one. The Area is then non-convex in π .

Now suppose all constrained axes form a prefix, but a non-singleton constrained axis precedes another constrained axis. Choose two values $x_1 < x_2$ inside the earlier interval and values z_{in} and z_{out} respectively inside and outside the later constraint. The Entries

$$(x_1, z_{in}), \quad (x_1, z_{out}), \quad (x_2, z_{in})$$

appear in this order while only the first and third satisfy the Area. Again the result is non-convex. Path cones are intervals by Lemma 1, so the same argument applies to the path axis. $\square$

Corollary 1 (Limits of one-entry coordinate indexes). *No lexicographic permutation of (S, P, T) makes either $P+T$ or $S+P+T$ Areas contiguous in general. $S+P$ is contiguous only when S precedes P and both precede the remaining axis; $S+T$ analogously requires an ST -leading order. Consequently:*

1. *no single coordinate permutation makes all eight Area shapes contiguous;*
2. *the SPT/PTS/TSP combination makes exactly FULL, S, P, T, and S+P uniformly contiguous;*
3. *even all six coordinate permutations together still fail to make $P+T$ and $S+P+T$ uniformly contiguous.*

The corollary is stronger than a criticism of one particular TripleStore permutation. Within the one-representative-per-index model, there is no missing permutation that completes the design.

Non-contiguity also implies a worst-case cost issue. The three-point counterexamples in the proof can be repeated across independent leading-coordinate values, producing arbitrarily many alternating inside/outside groups. Exact retrieval can therefore require $\Theta(N)$ disjoint fragments, while replacing those fragments with one bounding interval can produce $\Theta(N)$ false-positive scans.

The limitation disappears only by leaving this representation class: either use a genuinely multidimensional access method, or change what one logical Entry contributes to the ordered index. The next section takes the second route.

5 AreaScopeIndex: Materializing Area Scope Membership

AreaScopeIndex changes the representation rather than adding another coordinate permutation. Its construction is a denormalization of a derived relation: whether a logical Entry belongs to a given Area scope. The non-temporal Area predicate is materialized at write time; the temporal predicate remains an ordinary interval evaluated inside that scope.

For an Area scope $\gamma = (\alpha, p)$, an Entry e belongs to γ exactly when

$$s(e) \in \mathcal{S}_\alpha \quad \text{and} \quad p \preceq p(e).$$

Because the subspace mode is exact-or-ANY and the path predicate is prefix membership, the complete set of scopes containing one Entry can be enumerated directly from that Entry.

5.1 Materialized Area scopes

Let

$$e = (s, q, t, \dots)$$

be an Entry. For every

$$p \in \text{Pref}(q)$$

create two scope identifiers: the unrestricted-subspace scope

$$g_p^{\text{ANY}} = \langle \text{ANY}, p \rangle,$$

and the exact-subspace scope

$$g_p^s = \langle s, p \rangle.$$

The symbol ANY is a tagged index value and cannot collide with a valid SubspaceId.

The scope set of e is

$$G(e) = \{g_p^{\text{ANY}}, g_p^s \mid p \in \text{Pref}(q)\}. \quad (4)$$

For path depth d ,

$$|G(e)| = 2(d + 1). \quad (5)$$

This is the essential denormalization step. Every logical Entry is represented once in every Area scope to which it belongs. No time interval is materialized. Instead, all Entries of each scope share one canonical ordered sequence.

5.2 Canonical within-scope order

Let $\tau(e) \in \mathcal{U}$ be any peer-independent deterministic tie-breaker that totally orders distinct live Entries sharing the same timestamp. Define

$$\kappa(e) = (t(e), \tau(e)), \quad (6)$$

ordered lexicographically. For the formal model, the particular representation of τ is inessential: it may be the remaining Willow coordinates, a stable unique identifier, or another canonical value, provided that the resulting order is deterministic and injective on the live Entry set. When a probabilistic identifier such as a hash is used, the usual collision assumption applies.

The complete AreaScopeIndex key is only

$$(g, \kappa(e)). \quad (7)$$

Choose any deterministic total order among scope identifiers. The leading scope coordinate makes every materialized Area scope one contiguous block. Timestamp is the first component of κ , so every such block is timestamp-ordered. Let $\perp$ denote a sentinel smaller than every valid tie-breaker value; then $(t, \perp)$ denotes the lower boundary of timestamp t independently of how ties are represented.

One convenient concrete instantiation is $\tau(e) = (p(e), s(e))$. This form is used only where a coordinate-level example is useful; none of the Area contiguity or B+-tree results depends on this particular tie-breaker.

5.3 Insertion and deletion

Index representatives are a deterministic function of the Entry itself.

Listing 1: Generating all AreaScopeIndex representatives for one Entry.

```
INDEX_ENTRIES(e):
    k = KAPPA(e)
    for p in PREFIXES(e.path):      # includes the empty path
        emit (scope = (ANY, p),      key = k) -> reference(e)
        emit (scope = (e.subspace, p), key = k) -> reference(e)
```

Insertion and deletion are symmetric.

Listing 2: Insertion and deletion in AreaScopeIndex.

```
INSERT(e):
    STORE.insert(e)
    for i in INDEX_ENTRIES(e):
        INDEX.insert(i.key, reference(e))

DELETE(e):
    for i in INDEX_ENTRIES(e):
        INDEX.delete(i.key)
    STORE.delete(e)
```

No side table listing secondary representatives is required: their keys can be regenerated from the logical Entry. The payload need not be duplicated. An index representative can contain only its key, the contribution required by cached summaries, and a primary-record reference.

Willow prefix pruning can cause one application write to remove several pre-existing logical Entries. Equation (5) and the update bounds below are therefore per inserted or deleted live Entry; the number of Entries pruned by one application-level write is orthogonal to the index construction.

5.4 Worked example: explicit index layout

Consider the five logical Entries in Table 3. Alice and Bob are SubspaceIds. For this example use the concrete tie-breaker $\tau(e) = (p(e), s(e))$. Records r_2 and r_4 intentionally share timestamp 14, so the tie-breaker is visible in the index order. For later queries use

$$I = [12, 23).$$

Table 3: Example logical Entries.

Record	Subspace	Path	Timestamp
r_1	Alice	/Alice	10
r_2	Bob	/Alice/photos	14
r_3	Alice	/Bob	18
r_4	Alice	/Alice/docs	14
r_5	Bob	/Alice/notes	24

The five Entries form a valid Willow store under prefix pruning. In particular, the two Bob paths /Alice/photos and /Alice/notes are incomparable, so the newer r_5 does not prune r_2 .

For example,

$$\text{Pref}(/Alice/photos) = \{\epsilon, /Alice, /Alice/photos\}.$$

Hence r_2 produces six representatives: three ANYscopes and three Bob scopes.

For concreteness, assume all ANYscopes precede exact-subspace scopes; within each category scopes are ordered lexicographically by path; exact subspaces are ordered Alice before Bob. Table 4 gives the complete physical index. Labels $i_1, \dots, i_{26}$ name index representatives, not logical records.

Table 4: Complete AreaScopeIndex layout for Table 3.

Index entry	Scope	Time	Full path	Record
i_1	$\langle \text{ANY}, \epsilon \rangle$	10	/Alice	r_1
i_2	$\langle \text{ANY}, \epsilon \rangle$	14	/Alice/docs	r_4
i_3	$\langle \text{ANY}, \epsilon \rangle$	14	/Alice/photos	r_2
i_4	$\langle \text{ANY}, \epsilon \rangle$	18	/Bob	r_3
i_5	$\langle \text{ANY}, \epsilon \rangle$	24	/Alice/notes	r_5
i_6	$\langle \text{ANY}, \text{/Alice} \rangle$	10	/Alice	r_1
i_7	$\langle \text{ANY}, \text{/Alice} \rangle$	14	/Alice/docs	r_4
i_8	$\langle \text{ANY}, \text{/Alice} \rangle$	14	/Alice/photos	r_2
i_9	$\langle \text{ANY}, \text{/Alice} \rangle$	24	/Alice/notes	r_5
i_{10}	$\langle \text{ANY}, \text{/Alice/docs} \rangle$	14	/Alice/docs	r_4
i_{11}	$\langle \text{ANY}, \text{/Alice/notes} \rangle$	24	/Alice/notes	r_5
i_{12}	$\langle \text{ANY}, \text{/Alice/photos} \rangle$	14	/Alice/photos	r_2
i_{13}	$\langle \text{ANY}, \text{/Bob} \rangle$	18	/Bob	r_3
i_{14}	$\langle \text{Alice}, \epsilon \rangle$	10	/Alice	r_1
i_{15}	$\langle \text{Alice}, \epsilon \rangle$	14	/Alice/docs	r_4
i_{16}	$\langle \text{Alice}, \epsilon \rangle$	18	/Bob	r_3
i_{17}	$\langle \text{Alice}, \text{/Alice} \rangle$	10	/Alice	r_1
i_{18}	$\langle \text{Alice}, \text{/Alice} \rangle$	14	/Alice/docs	r_4
i_{19}	$\langle \text{Alice}, \text{/Alice/docs} \rangle$	14	/Alice/docs	r_4
i_{20}	$\langle \text{Alice}, \text{/Bob} \rangle$	18	/Bob	r_3
i_{21}	$\langle \text{Bob}, \epsilon \rangle$	14	/Alice/photos	r_2
i_{22}	$\langle \text{Bob}, \epsilon \rangle$	24	/Alice/notes	r_5
i_{23}	$\langle \text{Bob}, \text{/Alice} \rangle$	14	/Alice/photos	r_2
i_{24}	$\langle \text{Bob}, \text{/Alice} \rangle$	24	/Alice/notes	r_5
i_{25}	$\langle \text{Bob}, \text{/Alice/notes} \rangle$	24	/Alice/notes	r_5
i_{26}	$\langle \text{Bob}, \text{/Alice/photos} \rangle$	14	/Alice/photos	r_2

The block $i_6, \dots, i_9$ contains exactly all logical Entries in the cone of /Alice, independent of subspace, already sorted by timestamp. The block i_{17}, i_{18} contains exactly the same path cone restricted to Alice.

Table 5 applies all eight query shapes to this one AreaScopeIndex layout. Constrained path queries use /Alice; constrained subspace queries use Alice; constrained time queries use [12, 23).

Table 5: All eight Area shapes as contiguous ranges of the explicit AreaScopeIndex layout.

Shape	Scope	Time	Index interval	Records
FULL	$\langle \text{ANY}, \epsilon \rangle$	all	$i_1:i_5$	r_1, r_4, r_2, r_3, r_5
T	$\langle \text{ANY}, \epsilon \rangle$	[12, 23)	$i_2:i_4$	r_4, r_2, r_3
S	$\langle \text{Alice}, \epsilon \rangle$	all	$i_{14}:i_{16}$	r_1, r_4, r_3
P	$\langle \text{ANY}, \text{/Alice} \rangle$	all	$i_6:i_9$	r_1, r_4, r_2, r_5
S+P	$\langle \text{Alice}, \text{/Alice} \rangle$	all	$i_{17}:i_{18}$	r_1, r_4
S+T	$\langle \text{Alice}, \epsilon \rangle$	[12, 23)	$i_{15}:i_{16}$	r_4, r_3
P+T	$\langle \text{ANY}, \text{/Alice} \rangle$	[12, 23)	$i_7:i_8$	r_4, r_2
S+P+T	$\langle \text{Alice}, \text{/Alice} \rangle$	[12, 23)	i_{18}	r_4

Because materialization is performed over Willow Path components, a record at /Alice1 would never receive a representative under the scope for /Alice. Thus, for example, /Alice is an ancestor of /Alice/photos, but not of /Alice1.

5.5 Contiguity theorem

Lemma 2 (Scope membership). *Let e be an Entry and $A = (\alpha, p, I)$ an Area. Define*

$$g(A) = \begin{cases} \langle \text{ANY}, p \rangle, & \alpha = \text{ANY}, \\ \langle \alpha, p \rangle, & \alpha \in \mathcal{S}. \end{cases}$$

Then e has exactly one `AreaScopeIndex` representative in scope $g(A)$ if and only if it satisfies the subspace and path predicates of A .

Proof. If $\alpha = \text{ANY}$, a representative under $\langle \text{ANY}, p \rangle$ exists exactly when $p \in \text{Pref}(p(e))$, which by Equation (1) is equivalent to $p \preceq p(e)$. If $\alpha = s$, the exact-subspace representative exists exactly when $s(e) = s$ and $p \preceq p(e)$. Each prefix occurs once in $\text{Pref}(p(e))$, hence the representative is unique. $\square$

Theorem 1 (AreaScopeIndex contiguity). *For every Willow Area*

$$A = (\alpha, p, [t_0, t_1]),$$

all and only its Entries have one representative in the contiguous `AreaScopeIndex` range

$$[(g(A), (t_0, \perp)), (g(A), (t_1, \perp))]. \quad (8)$$

Open-ended time ranges use the end sentinel of the scope block.

Proof. Lemma 2 reduces the subspace and path predicates to equality on the single leading scope value $g(A)$. All representatives of this scope are contiguous. Within the scope, timestamp is the first component of κ , so $t_0 \leq t < t_1$ selects one contiguous subrange. Every logical Entry satisfying the Area predicates has exactly one representative there, and no other Entry does. $\square$

Corollary 2 (Uniformity of all Area shapes). *All eight shapes `FULL`, `T`, `S`, `P`, `S+P`, `S+T`, `P+T`, and `S+P+T` reduce to choosing one materialized Area scope and one timestamp interval. The access path is therefore uniform across shapes, with no query-time filtering or interval decomposition.*

Theorem 1 contains no path-depth assumption. Path depth affects only the number of maintained representatives.

6 B+-Tree Consequences and Maintenance Cost for AreaScopeIndex

AreaScopeIndex requires only a total order, so a conventional B+-tree is sufficient. To support range summarization and balanced splitting, each internal child reference can cache a fixed-size composable summary and the number of descendant representatives.

Let

$$\mathcal{G}(X) = (|X|, \Sigma(X))$$

be the cached aggregate of a subtree range X , where Σ is composable under an associative operation. This is the range-summarizable order-statistics pattern: cached counts support rank/select, and cached summaries support arbitrary contiguous-range aggregation [1].

6.1 Area lookup, enumeration, and summarization

Let h be B+-tree height. By Theorem 1, locating an Area requires ordinary searches for the two boundaries of one interval. Enumeration then scans leaves sequentially. For m returned Entries,

$$\text{Enumerate}(A) = O(h + m).$$

Range summarization descends only along the two boundary regions; fully covered subtrees strictly between them contribute cached summaries directly. Under bounded page size,

$$\text{Summary}(A) = O(h).$$

Similarly, locating the interval costs $O(h)$.

Thus

$$\boxed{\text{Locate}(A) = O(h), \quad \text{Summary}(A) = O(h), \quad \text{Enumerate}(A) = O(h + m).}$$

The practically relevant statement is not constant RAM-model time, but a *fixed number of ordinary B+-tree traversals independent of Area shape and Area cardinality*. As usual, high B+-tree fanout makes h small in persistent stores.

6.2 Rank, select, and balanced cuts

Cached subtree counts provide ordinary order statistics. $\text{Rank}(x)$ follows one root-to-leaf search while accumulating counts of preceding siblings; $\text{Select}(r)$ chooses the child containing the residual rank. Both cost $O(h)$ [1].

For one contiguous interval $[l, u)$ containing m representatives, a balanced b -way split can be constructed by

$$r_0 = \text{Rank}(l),$$

$$q_j = \left\lfloor \frac{jm}{b} \right\rfloor, \quad c_j = \text{Select}(r_0 + q_j), \quad j = 1, \dots, b-1.$$

With $c_0 = l$ and $c_b = u$, the resulting subranges differ in cardinality by at most one. For fixed b ,

$$\text{Split}_b = O(bh) = O(h).$$

This is exactly the balanced splitting primitive needed by range-based set reconciliation [10, 1].

The timestamp-first order also matches Willow’s explicit `AreaOfInterest.max_count` requirement, which restricts an interest to Entries among the newest in a store [16]. A recency restriction selects a suffix of the canonical timestamp-first order; intersecting that suffix with an Area scope is therefore a slice. With deterministic tie handling through κ , the boundary for the selected newest Entries can be located by rank/select in $O(h)$. The independent `max_size` condition may additionally benefit from cached payload-length aggregates, but is not analyzed here.

6.3 Space and update amplification

An Entry at path depth d has $d + 1$ prefixes, each represented once in the exact-subspace scope and once in the ANYscope. Hence

$$M(d) = 2(d + 1)$$

index representatives are created exactly.

Assume an effective maximum path depth

$$d \leq D.$$

Then

$$M(d) \leq 2(D + 1),$$

and for N logical Entries the representative count is

$$O(ND).$$

Each representative insertion or deletion is one ordinary B+-tree update. With tree height h , per-record maintenance is

$$O(Dh),$$

with at most $2(D + 1)$ root-to-leaf update operations. Aggregate maintenance is included in this bound because counts and summaries are refreshed bottom-up along the affected paths and any ordinary rebalancing cascade [1].

At the level of index memberships,

Representation	Memberships per logical Entry
Single SPT store	1
TripleStore	3
AreaScopeIndex	$2(d + 1)$

`AreaScopeIndex` is therefore not an unconditional storage reduction. It exchanges bounded write and space amplification for uniform exact reads. This trade-off is particularly plausible in replication- and reconciliation-heavy deployments: one durable write may subsequently be read by many peers, while one reconciliation session can issue many range-summary, rank/select, and residual-enumeration queries. In such workloads, bounded work paid once on update can amortize over substantially more read-side operations. The assumption is asymmetric and should be made explicit rather than treated as universally favorable.

Representative count also differs from byte-level amplification. A straightforward key encoding repeats prefix bytes across representatives, with total prefix material proportional to

$$\sum_{p \in \text{Pref}(q)} |p|.$$

Willow bounds both path depth and total path length [15, 13]; practical B+-trees can additionally apply page-level prefix compression. No claim is made that the resulting physical byte cost is always smaller than TripleStore.

7 Range Reconciliation and the 3D Range Language

The one-dimensional storage representation can be exploited most directly if Willow’s current reconciliation range language is changed slightly.

Willow adapts range-based set reconciliation to three-dimensional data by delimiting reconciliation sets with 3dRanges [17]. A peer receiving a mismatching fingerprint may reply with fingerprints whose 3dRanges cover the parent range. Balanced refinement is recommended to follow Entry cardinality rather than geometric volume.

This formulation is explicitly storage-order agnostic. The Willow RBSR specification requires the fingerprint-combination operation to be commutative because it does not wish to prescribe how three-dimensional data are linearized [17]. Under that premise, generic boxes are a natural common language: both peers can identify the same subset without agreeing on one ordered representation.

Once Theorem 1 supplies a canonical efficient order for every Area, the trade-off changes. Reconciliation initially concerns an Area obtained from the peers’ interests, but balanced rank cuts of that Area are generally not Areas. The current protocol therefore leaves the efficient Area language as soon as arbitrary balanced refinement is required.

For a coordinate-level illustration, instantiate the otherwise abstract tie-breaker as $\tau(e) = (p(e), s(e))$, hence $\kappa(e) = (t(e), p(e), s(e))$. Consider

$$A = (\text{ANY}, /b\log, T)$$

and a cut at canonical key

$$c = (14, /b\log/c, \text{Bob}).$$

The lower ordered part of A is

$$\begin{aligned} \{e \in A \mid \kappa(e) < c\} = & \{T < 14, P \in \mathcal{C}(/b\log)\} \\ & \cup \{T = 14, P \in \mathcal{C}(/b\log), P < /b\log/c\} \\ & \cup \{T = 14, P = /b\log/c, S < \text{Bob}\}. \end{aligned}$$

This set is one contiguous interval of the canonical Area order, but it is neither an Area nor one generic 3D box. It is naturally representable as a small union of boxes.

Three observations follow.

1. A balanced split is trivial in the ordered index but can become several 3dRanges on the wire.
2. A cut key alone does not encode the parent Area’s exact-subspace/ANYchoice and prefix cone; this scope must remain part of the range description.
3. The resulting boxes are often not Areas, so a backend designed only for efficient Areas can be forced back into generic multidimensional processing during reconciliation.

Restricting cuts to timestamp boundaries avoids the third issue but loses balanced divisibility when many Entries share one timestamp. In the extreme case where all Entries have the same timestamp, no nontrivial timestamp split exists. Rank/select over the canonical key has no such limitation because path and subspace break timestamp ties.

Thus the current range language presents a tension:

Areas: storage-friendly but not closed under the balanced splitting required by RBSR
generic 3dRanges: expressive enough for refinement but not uniformly storage-friendly.

The next section introduces a range language that is closed under the operations needed by reconciliation while every term remains one native AreaScopeIndex interval. After eliminating multidimensional access from the store, such a language follows the native representation more directly than arbitrary coordinate boxes.

8 Slices: A Closed Ordered Range Language

8.1 Definition

Let $\mathcal{K}$ denote the totally ordered range of canonical within-scope keys $\kappa(e)$. For open-ended ranges, extend it with the usual boundary sentinels and write the extended domain as $\overline{\mathcal{K}}$.

Definition 4 (Slice). A slice is a triple

$$\sigma = (\alpha, p, [k, k']),$$

where $\alpha \in \mathcal{S} \cup \{\text{ANY}\}$, $p \in \mathcal{P}$, and $k, k' \in \overline{\mathcal{K}}$. Its denotation is

$$\mathcal{L}(\sigma) = \{e \mid s(e) \in \mathcal{S}_\alpha \wedge p(e) \in \mathcal{C}(p) \wedge k \leq \kappa(e) < k'\}. \quad (9)$$

The pair $\langle \alpha, p \rangle$ is the slice scope and $[k, k')$ its canonical-key interval.

A slice retains exactly the information that a cut key alone lacks: the subspace selection and prefix cone of the parent, together with an interval of the canonical order inside that scope.

Lemma 3 (Areas are slices). Every Area $A = (\alpha, p, [t_0, t_1))$ is the slice

$$\sigma_A = (\alpha, p, [(t_0, \perp), (t_1, \perp))).$$

The finite upper key is replaced by the corresponding open-end sentinel when $t_1 = \infty$.

Proof. $(t_0, \perp)$ is the least canonical key with timestamp t_0 . Therefore the canonical-key interval contains exactly the keys with $t_0 \leq t < t_1$ inside the fixed Area scope $\langle \alpha, p \rangle$. $\square$

The inclusion is strict: the balanced-cut example below gives slices that are not Areas. Thus

$$\text{Areas} \subsetneq \text{Slices}. \quad (10)$$

Together with Equation (3), this shows two different extensions of the Area language. Generic 3dRanges add coordinate-wise geometric freedom, whereas slices add freedom to cut the canonical order within a fixed Area scope. Neither range language contains the other in general.

Theorem 2 (Slice contiguity). For every slice $\sigma = (\alpha, p, [k, k'))$, all and only the Entries of $\mathcal{L}(\sigma)$ have one representative in the `AreaScopeIndex` interval

$$[(\langle \alpha, p \rangle, k), (\langle \alpha, p \rangle, k')).$$

Consequently, on a range-summarizable B+-tree, locating and summarizing a slice, and computing rank or select within it, cost $O(h)$; enumerating its m Entries costs $O(h + m)$.

Proof. Lemma 2 provides exactly one representative in scope $\langle \alpha, p \rangle$ for each Entry satisfying the slice's subspace and cone predicates. Equation (9) then restricts those representatives to one canonical-key interval. The B+-tree costs are the same as for Areas. $\square$

8.2 Example: a balanced cut inside one timestamp

The running example of Section 5.4 already contains the case that motivates slices. In scope

$$\langle \text{ANY}, / \text{Alice} \rangle,$$

the four representatives $i_6, \dots, i_9$ are ordered as

$$i_6 : r_1 @ 10, \quad i_7 : r_4 @ 14, \quad i_8 : r_2 @ 14, \quad i_9 : r_5 @ 24.$$

The two Entries at timestamp 14 are ordered by the concrete tie-breaker $\tau(e) = (p(e), s(e))$ used in the example, so r_4 precedes r_2 because `/Alice/docs` precedes `/Alice/photos`.

Consider the Area

$$A = (\text{ANY}, / \text{Alice}, [10, 25)),$$

which is exactly the `AreaScopeIndex` interval $i_6:i_9$. A balanced two-way split after two Entries places the cut between i_7 and i_8 , hence at

$$c = \kappa(r_2).$$

The two resulting slices are

$$\begin{aligned} \sigma_1 &= (\text{ANY}, / \text{Alice}, [(10, \perp), c)), \\ \sigma_2 &= (\text{ANY}, / \text{Alice}, [c, (25, \perp))). \end{aligned}$$

They correspond exactly to the contiguous index ranges

$$i_6:i_7 \quad \text{and} \quad i_8:i_9,$$

and contain respectively $\{r_1, r_4\}$ and $\{r_2, r_5\}$. The cut is perfectly balanced, but it cannot be represented by a timestamp boundary because it separates r_4 and r_2 , which share timestamp 14. A slice records precisely this within-scope cut while remaining a native `AreaScopeIndex` interval.

8.3 Closure under balanced splitting

Theorem 3 (Closure under splitting). *Let*

$$\sigma = (\alpha, p, [k, k'])$$

and let

$$k = c_0 < c_1 < \dots < c_b = k'.$$

Then

$$\sigma_j = (\alpha, p, [c_{j-1}, c_j]), \quad j = 1, \dots, b,$$

are slices that partition $\mathcal{L}(\sigma)$.

If a store contains $m \geq b$ Entries in $\mathcal{L}(\sigma)$ and the internal cut c_j is chosen at relative rank $\lfloor jm/b \rfloor$, the resulting parts contain either $\lfloor m/b \rfloor$ or $\lceil m/b \rceil$ local Entries, and all cuts are computed in $O(bh)$ B+-tree work.

Proof. The key intervals partition $[k, k']$ while the scope is unchanged. Since κ is injective on the live store, Entries in the slice are totally ordered without ties. The rank/select construction of Section 6.2 yields the stated balanced cardinalities and cost. $\square$

Unlike timestamp-only cuts, Theorem 3 remains exact when many Entries have the same timestamp because τ provides a deterministic within-timestamp order.

8.4 Closure under intersection

Proposition 2 (Slice intersection). *Let*

$$\sigma_i = (\alpha_i, p_i, K_i), \quad i \in \{1, 2\},$$

where K_i is a canonical-key interval. Their intersection is either empty or the slice

$$(\alpha_1 \sqcap \alpha_2, p_1 \sqcup p_2, K_1 \cap K_2),$$

where ANY $\sqcap \alpha = \alpha$, equal exact subspaces intersect to themselves, distinct exact subspaces give the empty set, and $p_1 \sqcup p_2$ is the longer prefix when one of p_1, p_2 prefixes the other; unrelated cones give the empty set.

Proof. Slice membership is the conjunction of one subspace predicate, one cone predicate, and one canonical-key interval predicate. Singleton/all subspace predicates are closed under conjunction. Prefix cones are either nested or disjoint, so their intersection is the narrower cone or empty. Intersections of intervals in one total order are intervals or empty. $\square$

This property is useful for capability- and interest-bounded reconciliation because the authorization or interest scope remains explicit while the local range stays one `AreaScopeIndex` interval.

8.5 Slice-based reconciliation

A reconciliation session is *slice-conforming* if it begins from Areas and refines every mismatching range only by balanced slice splits from Theorem 3. Note that this is a protocol variant, not the current Willow reconciliation protocol. Two peers must (i) agree on the same canonical within-scope order κ , including its tie-breaking semantics, and (ii) exchange slice descriptors carrying the Area scope and the canonical interval $[k, k']$. The latter requires a range-message format for slices rather than only the current Willow `3dRange` descriptor.

The local storage assumption is also explicit: each peer stores `AreaScopeIndex` in a range-summarizable B+-tree of height h , with cached subtree cardinalities and composable summaries as in Section 6.

Theorem 4 (Native reconciliation invariant). *Under the shared- κ , slice-message, and storage assumptions above, every logical range in a slice-conforming reconciliation session is one contiguous `AreaScopeIndex` interval. For fixed split fanout b , every range summary costs $O(h)$ and every balanced split costs $O(bh)$; enumerating a residual range containing m Entries costs $O(h + m)$. On finite stores, if mismatching ranges below a fixed cardinality threshold are enumerated rather than split, the recursive reconciliation terminates.*

Proof. The initial reconciliation range is an Area and therefore a slice by Lemma 3. Every recursive child is a slice by Theorem 3. Every slice is one `AreaScopeIndex` interval by Theorem 2. The cost bounds follow from the range-summarizable B+-tree operations of Section 6. For termination, every nonterminal balanced split introduces strict child subintervals at canonical keys, while a small residual range is sent explicitly. Since the union of the two finite peer stores contains only finitely many canonical keys, an infinite strictly descending chain of such refinements is impossible. $\square$

Slices therefore provide one language for both the initial query-derived reconciliation set and all subsequent balanced splits. The relationship between slices and Willow 3dRanges is analyzed separately in Appendix A.

9 Related Work

9.1 Ancestor materialization and hierarchical search

The denormalization techniques used by `AreaScopeIndex` are not new in isolation: ancestor materialization, materialized paths, and hierarchical token expansion are established ways of exchanging additional write/storage work for simpler descendant queries. What is specific here is their combination with Willow’s exact-or-any subspace dimension, timestamp ordering, and range summarization.

The closest structural precedent I am aware of is Google Cloud Datastore’s ancestor-aware indexing. Ancestor-aware composite indexes conceptually include one index position for each relevant ancestor relation [6]. This is closely related to materializing $\text{Pref}(p)$, but Datastore does not expose Willow’s fixed combination of exact-or-any subspace selection, hierarchical path cones, temporal intervals, and range summarization.

MongoDB’s documented *Array of Ancestors* pattern stores ancestor identifiers with each document and indexes that array for descendant queries [11]. This captures the same hierarchy-versus-write-amplification trade-off within a generic document model.

Lucene’s `PathHierarchyTokenizer` performs almost exactly the local path expansion used here: a hierarchical path is tokenized into its successive prefixes [2]. Lucene uses the resulting terms in an inverted index; the present construction combines each prefix with subspace scope and timestamp to obtain ordered, summarizable ranges.

These techniques are also related to classical materialized-path and closure-table representations of hierarchies [3]. The distinctive point here is not ancestor materialization by itself, but that Willow’s restricted Area semantics let it eliminate the entire multidimensional query at the storage layer.

9.2 Materializing an all-values dimension

The subspace-independent scope is analogous to the ALL value of data cubes [7]. The pairs

$$\langle s, p \rangle \quad \text{and} \quad \langle \text{ANY}, p \rangle$$

materialize both a concrete subspace and a view independent of subspace value. Unlike a classical cube cell, however, ANY materializes membership in an ordered sequence rather than only one aggregate; enumeration, rank/select, temporal subranges, and summaries remain available.

9.3 Controlled redundancy for multidimensional access

Redundancy has long been used to improve multidimensional indexing. Orenstein studied controlled redundancy in spatial databases as a trade-off between extra index representation and false hits/I/O [12]. Z-order and related methods attempt to approximately preserve multidimensional locality by mapping the original coordinates into one scalar order, without denormalizing semantic query membership. `AreaScopeIndex` takes the opposite route: it denormalizes the restricted Area predicates themselves so that every supported query becomes an exact interval. This also differs from the Z-order direction mentioned in Willow’s own storage notes and `willow-js` roadmap [14, 5].

9.4 Range reconciliation over ordered stores

Negentropy is the other practical RBSR implementation considered here, operating over records in a canonical timestamp-and-identifier order [8]. The Nostr relay `strfry` can cache Negentropy B-trees for the full database and selected configured filters [9].

Conceptually, `AreaScopeIndex` performs a reduction from Willow’s restricted three-dimensional Area model to a denormalized family of canonical one-dimensional sequences. After this reduction, the local storage problem has the same essential primitives as Negentropy: summarize one ordered range, subdivide it, and enumerate small residual ranges. Willow still has an additional hierarchical query boundary: a reconciliation starts by selecting an Area scope, and recursive splits must retain that scope. Slice is the natural language that joins these two roles, the query-derived range and the split range, without returning to multidimensional storage.

The range-summarizable order-statistics view of RBSR makes explicit that arbitrary contiguous-range summaries, rank, select, and enumeration are sufficient local primitives for balanced reconciliation [1]. `AreaScopeIndex` supplies exactly such ranges for Willow Areas, while slices keep recursive refinement in the same one-dimensional model.

10 Discussion

The observations above suggest that Willow’s apparent three-dimensionality can be separated into two layers. At the data-model level, subspace, path, and timestamp remain three distinct coordinates with meaningful semantics. At the storage level, however, the restricted Area language permits a denormalized reduction to one-dimensional ordered scope ranges. `AreaScopeIndex` performs that reduction explicitly: the exact-or-ANYsubspace choice and the path-prefix relation are paid for at write time, leaving one canonical ordered interval at read time.

This changes the interpretation of the reconciliation problem as well. Once an Area is one ordered range, slices are the minimal extension needed for arbitrary balanced cuts: they preserve the Area scope and vary only the interval of the canonical order. In that sense, Slice is not merely an additional range type; it is the form that aligns Willow’s query language with the range language required by ordered reconciliation.

Fingerprint commutativity becomes a trade-off rather than a geometric requirement. The Willow RBSR specification requires an associative and commutative fingerprint combination because it deliberately leaves the linearization of three-dimensional data unspecified. It explicitly notes that the non-commutative Cayley hashes surveyed by Meyer are therefore excluded, whereas the commutative candidates remain suitable [17, 10]. A shared canonical order κ removes this *particular* reason for commutativity: secure order-sensitive tree-friendly functions, including the non-commutative families analyzed for one-dimensional RBSR, become candidates again.

This does not make commutative algebra undesirable. Negentropy, despite operating on a canonical timestamp-and-identifier order, deliberately uses addition modulo 2^{256} of record identifiers (plus range cardinality and final hashing), hence a commutative incremental fingerprint [8]. Moreover, delta-style maintenance when a `LengthyAuthorisedEntry`’s available payload prefix advances benefits from an invertible combine operation: replacing an old singleton contribution by a new one through a delta requires group-like structure. The canonical order therefore broadens the admissible fingerprint designs, but the choice among commutative groups, non-commutative tree-friendly functions, and other summaries remains an implementation trade-off involving security, update cost, and range-summary performance.

Depth becomes an explicit systems parameter. In `AreaScopeIndex`, the read-side uniformity is purchased with $2(d+1)$ representatives per live Entry. Willow already bounds component count, but the Willow’25 protocol-level maximum is intentionally generous [13]. Under `AreaScopeIndex`, effective path depth becomes a direct storage-design parameter because it controls write and space amplification. Distinguishing the protocol maximum from a smaller workload- or application-level effective bound D therefore becomes operationally important. The contiguity theorem itself does not depend on this bound; only the maintenance bound does.

Bounded structural work reduces shape-based adversarial amplification. The coordinate-index layouts analyzed in Section 3 have legal Area shapes whose local work is data-dependent and can approach a linear scan or a linear number of fragments. When clients can choose Areas, this creates an obvious systems concern: pathological but valid query shapes can concentrate server work even when the returned set is small. Under `AreaScopeIndex`, the structural work of a range summary, rank, or select is $O(h)$ regardless of Area shape, and an insertion or deletion is $O(Dh)$ under an effective depth bound D . With fixed D and the high fanout typical of persistent B+-trees, these bounds correspond to a small, predictable number of tree traversals. They do not bound the number of client requests or the unavoidable cost of enumerating a large result, so ordinary admission control and protocol-level resource limits remain necessary. But the benefit is clear: Area geometry itself no longer provides a source of potentially large server-side work.

The trade-off is intentionally asymmetric. The representation favors repeated reads, summaries, and reconciliation over minimal write amplification. This is well matched to deployments in which one accepted write may be observed by many readers or peers, and in which one reconciliation session evaluates multiple nested ranges before converging. It is less compelling for write-dominated workloads with very deep paths. The appropriate D and physical key representation are therefore empirical systems choices even though the contiguity result is purely structural.

Overall, the analysis points to a simpler conceptual chain:

Willow Area $\longrightarrow$ denormalized canonical 1D range $\longrightarrow$ Slice refinement.

This should be read as an area for further design analysis rather than as a prescribed protocol transition. The main formal result is narrower: the restricted semantics of Areas are sufficient to remove query-shape-dependent multidimensional access from the storage layer.

11 Conclusion

Willow data are naturally described by three coordinates, but efficient Willow Area access does not need to remain a generic three-dimensional indexing problem. Areas expose a much narrower predicate language than arbitrary boxes: subspace is exact-or-ANY, path is a prefix cone, and time is one interval.

TripleStore and PersistentStore illustrate two current coordinate-index strategies. The first maintains several lexicographic orders; the second uses one SPT order and filters bounding scans when a query does not align with it. The lexicographic contiguity result shows that this is not merely a matter of choosing more permutations. In the one-representative-per-index model used by these implementations, P+T and S+P+T cannot be made uniformly contiguous by any coordinate permutation, even if all six permutations are maintained.

AreaScopeIndex changes the representation. It materializes the derived membership relation between Entries and Area scopes: every ancestor path prefix is represented once under the Entry’s exact subspace and once under ANY. Within each scope, Entries are placed in a timestamp-first canonical order κ . An Area is therefore exactly one scope plus one timestamp interval. By Theorem 1, this makes the complete Area language contiguous in the same index, including the P+T and S+P+T shapes that no coordinate permutation can cover uniformly. The result is independent of path-depth bounds. A maximum effective depth D is needed only to bound the price at $2(D + 1)$ representatives and $O(Dh)$ B+-tree maintenance work per logical Entry update.

The same one-dimensional model simplifies the local structure needed by range reconciliation. A balanced cut of an Area need not be another Area, but it is always an interval of the canonical within-scope order. Slices make that interval explicit while retaining its Area scope. A slice-based protocol requires peers to share κ and to transmit slice descriptors; under those assumptions, a range-summarizable B+-tree keyed by AreaScopeIndex supports $O(h)$ summaries and rank/select navigation, output-sensitive enumeration, and balanced recursive splitting without returning to shape-dependent multidimensional access.

The resulting picture is a reduction from Willow’s restricted 3D Area model to a denormalized canonical 1D storage model. This brings its local reconciliation requirements close to those of ordinary ordered RBSR systems such as Negentropy, while Slice provides the additional language needed to preserve Willow’s hierarchical Area scope across recursive splits. The remaining design questions are therefore no longer whether efficient uniform Area access is possible, but how the write/read trade-off, path depth, summary algebra, and reconciliation language should be balanced around that representation.

Appendix

A Slices and Willow 3dRanges

The core AreaScopeIndex and Slice results do not require a coordinate-specific tie-breaker. This appendix studies the relationship with Willow’s 3dRange language under the convenient instantiation

$$\kappa(e) = (t(e), p(e), s(e)),$$

i.e. $\tau(e) = (p(e), s(e))$. Let s_{min} be the least SubspaceId. Under this instantiation, a slice interval can be decomposed directly along timestamp, path, and subspace coordinates.

Call a 3dRange *canonical* if it has one of the following forms:

- (C1) $\mathcal{S}_\alpha \times \mathcal{C}(p) \times I$: an Area;
- (C2) $\mathcal{S}_\alpha \times J_P \times \{t\}$: one subspace mode, one arbitrary path interval, one timestamp;
- (C3) $J_S \times \{q\} \times \{t\}$: one subspace interval, one path, one timestamp.

Each canonical box is itself a slice. C1 follows from Lemma 3. For C2,

$$\mathcal{S}_\alpha \times [a, b] \times \{t\} = \mathcal{L}(\alpha, \epsilon, [(t, a, s_{min}), (t, b, s_{min})]).$$

For C3,

$$[s_1, s_2] \times \{q\} \times \{t\} = \mathcal{L}(\text{ANY}, \epsilon, [(t, q, s_1), (t, q, s_2)]).$$

Theorem 5 (A slice is at most five canonical boxes). *Under the coordinate instantiation above, every slice*

$$\sigma = (\alpha, p, [k, k'])$$

can be expressed as the disjoint union of at most five canonical `3dRanges` computed from σ alone.

Proof. Write

$$k = (t, q, s), \quad k' = (t', q', s')$$

and let

$$B = \mathcal{S}_\alpha \times \mathcal{C}(p) \times \mathcal{T}.$$

Assume first $t < t'$. The lexicographic interval $[k, k']$ consists of five consecutive pieces:

$$\begin{aligned} Q_1 &= B \cap ([s, \infty) \times \{q\} \times \{t\}), \\ Q_2 &= B \cap (\mathcal{S} \times (q, \infty) \times \{t\}), \\ Q_3 &= B \cap (\mathcal{S} \times \mathcal{P} \times [t+1, t']), \\ Q_4 &= B \cap (\mathcal{S} \times [q, q'] \times \{t'\}), \\ Q_5 &= B \cap ([s_{min}, s'] \times \{q'\} \times \{t'\}). \end{aligned}$$

Empty pieces are omitted. Q_1 and Q_5 are of type C3, Q_2 and Q_4 of type C2, and Q_3 of type C1. If $t' = \infty$, the final two pieces disappear and the middle Area is open-ended. If $t = t'$, the analogous decomposition inside the (P, S) interval requires at most three boxes. The pieces are disjoint because they partition the lexicographic interval by timestamp, then path, then subspace. $\square$

The theorem is a relationship between the two range languages: slices are native intervals of `AreaScopeIndex`, while this particular coordinate order admits a bounded decomposition into Willow boxes. No such decomposition is required by the core contiguity or reconciliation results.

References

- [1] Elvio G. Amparore. Protocol-Storage Co-Design for Efficient Range-Based Set Reconciliation. In *The 24th IEEE International Symposium on Network Computing and Applications (NCA 2026)*, 2026. To appear in IEEE NCA 2026.
- [2] Apache Lucene. PathHierarchyTokenizer. Lucene 9.9 API documentation, 2024. URL https://lucene.apache.org/core/9_9_0/analysis/common/org/apache/lucene/analysis/path/PathHierarchyTokenizer.html. Inspected 2026-10-02.
- [3] Joe Celko. *Joe Celko's Trees and Hierarchies in SQL for Smarties*. Morgan Kaufmann, 2 edition, 2012. ISBN 978-0-12-387733-8.
- [4] Earthstar Project. Earthstar: Storage for Private, Distributed, Offline-First Applications. Software repository and project documentation, 2026. URL <https://github.com/earthstar-project/earthstar>. Repository states that v11 development uses Willow; inspected 2026-10-02.
- [5] Earthstar Project. willow-js / @earthstar/willow 0.6.1. Software repository and JSR package, 2026. URL <https://github.com/earthstar-project/willow-js>. JSR package version 0.6.1; main-branch source and README inspected 2026-10-02.
- [6] Google Cloud. Firestore in Datastore Mode: Indexes. Google Cloud documentation, 2026. URL <https://docs.cloud.google.com/datastore/docs/concepts/indexes>. See also the Datastore Admin API ALL_ANCESTORS option; inspected 2026-10-02.
- [7] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Technical Report MSR-TR-95-22, Microsoft Research, 1995. URL <https://www.microsoft.com/en-us/research/publication/data-cube-a-relational-aggregation-operator-generalizing-group-by-cross-tab-and-sub-totals/>.
- [8] Doug Hoyte and contributors. Negentropy: Range-Based Set Reconciliation. Software repository and protocol documentation, 2026. URL <https://github.com/hoytech/negentropy>. Inspected 2026-10-02.
- [9] Doug Hoyte and contributors. strfry: A Relay for the Nostr Protocol. Software repository, 2026. URL <https://github.com/hoytech/strfry>. Negentropy B-tree caching for configured filters; inspected 2026-10-02.

- [10] Aljoscha Meyer. Range-Based Set Reconciliation. *arXiv preprint arXiv:2212.13567*, 2023. URL <https://arxiv.org/abs/2212.13567>.
- [11] MongoDB. Model Tree Structures with an Array of Ancestors. MongoDB Database Manual, 2026. URL <https://www.mongodb.com/docs/v8.0/tutorial/model-tree-structures-with-ancestors-array/>. Inspected 2026-10-02.
- [12] Jack A. Orenstein. Redundancy in Spatial Databases. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 295–305. ACM, 1989. doi:10.1145/67544.66954. URL <https://doi.org/10.1145/67544.66954>.
- [13] Willow Project. Willow’25. Willow protocol parameter proposal, 2025. URL <https://willowprotocol.org/specs/willow25/>. Inspected 2026-10-02.
- [14] Willow Project. Three-Dimensional Data Storage. Willow Project technical notes, 2026. URL <https://willowprotocol.org/more/3dstorage/>. Inspected 2026-10-02.
- [15] Willow Project. Willow: Data Model. Willow Protocol Specification, 2026. URL <https://willowprotocol.org/specs/data-model/>. Inspected 2026-10-02.
- [16] Willow Project. Willow: Grouping Entries. Willow Protocol Specification, 2026. URL <https://willowprotocol.org/specs/grouping-entries/>. Inspected 2026-10-02.
- [17] Willow Project. 3d Range-Based Set Reconciliation. Willow Protocol Specification, 2026. URL <https://willowprotocol.org/specs/rbsr/>. Inspected 2026-10-02.
- [18] Willow Project contributors. willow25 0.7.9: PersistentStore source. docs.rs source for crate willow25, 2026. URL https://docs.rs/crate/willow25/0.7.9/source/src/storage/persistent_store.rs. Version 0.7.9 published 2026-08-27; source inspected 2026-10-02.