Person Alert Code

**Design Decisions**

*Proposed Sector-Wide Standard for Housing Providers (HACT/MHCLG aligned)*  
**Version:** Draft v1.1 | **Prepared by:** Elena Iurco | **Date:** 30 July 2025

# Overview

This module tracks alerts related to persons within a housing and social care context. Alerts capture critical health, safety, and risk information linked to tenants, household members, and staff, supporting housing providers to respond appropriately. The design aims to be:

* **Flexible**: Supports many types of alerts from various sources.
* **Normalized**: Avoids data duplication via lookup tables.
* **Traceable**: Keeps alert history and audit trails.
* **Secure and Consistent**: Uses foreign keys and constraints.

# Core Tables

## person\_alert

Purpose:  
The person\_alert table records individual alert instances associated with persons (tenants, household members, staff, etc.). Each alert corresponds to a specific alert code (e.g., health condition, risk, safeguarding concern) and includes contextual information such as the alert source, status, duration, and progress.

### Considerations:

* Captures multiple alert dimensions per person to support housing and social care needs assessment.
* Tracks alert provenance and lifecycle for auditability and case management.
* Enables linkage to other modules (e.g., repairs, social services) through a standardized person entity.
* Supports tenant safety by recording if the resident has been informed.
* Includes temporal data for managing alert validity and reviews.
* Designed with foreign key constraints to ensure referential integrity and support data consistency.

### Fields

#### person\_alert\_id

**What**: Unique identifier for each alert instance.  
**How is it designed**: INTEGER PRIMARY KEY AUTOINCREMENT to automatically generate unique IDs on insert.  
**How does it add value**: Enables efficient indexing, retrieval, and relationships with history or audit tables. Provides a stable reference for each alert instance.

**Questions to ask**:   
**What alternatives were considered**: None — this is fundamental for the table's function.

#### person\_id

**What**: Reference to the person (tenant, household member) the alert applies to.  
**How is it designed**: A mandatory foreign key (FK) to person(person\_id). Not nullable to enforce that every alert is linked to a valid person.  
**How does it add value**: Ties alerts directly to individuals for accurate data management.  
**Questions to ask**:  
**What alternatives were considered**: Could use separate fields for tenant and staff alerts, but this fragments data.

#### person\_alert\_code\_id

**What**: Reference to the standardised alert code from person\_alert\_codes.  
**How is it designed**: A mandatory foreign key. Not nullable to ensure valid alert categorization.  
**How does it add value**: Ensures consistent classification of alert types for analytics and action.  
**Questions to ask**:

* Should we allow local custom alert codes outside the core list?
* How flexible should alert codes be to allow new types?
* Is versioning of codes needed?

**What alternatives were considered**: Embedding codes directly as text, but that would undermine data consistency.

#### source\_type\_id

**What**: The origin of the alert (e.g. NHS, GP, Self-reported).  
**How is it designed**: Foreign key to the alert\_source\_types(source\_type\_id) lookup table. This field is optional to allow cases where the source might be unknown or not recorded.

**How does it add value**: Tracks provenance of information, useful for validation, follow-up, and audit trails.

**Questions to ask**:

* Should there be a default “Unknown” or “Not Specified” entry in the source\_types table to represent unknown sources?
* How frequently are new source types expected, and what governance process will manage them?

**What alternatives were considered**: Storing the source as free text, which can lead to inconsistent or ambiguous entries and complicates reporting and data quality assurance.

#### created\_by\_user\_id

**What:** Reference to the person who created the alert record – usually a staff member.

**How is it designed:** A foreign key pointing to the person\_id field in the **person** table.

**How does it add value:** Improves data integrity, reduces errors from manual name entry, and allows for role-based access control and audit logging across the system. Supports structured reporting on staff actions and workload.

**Questions to ask:**

* Should inactive or deleted users still be traceable in historic alert records?
* Should roles or permissions be enforced at the database level or in application logic?

**What alternatives were considered:** Free-text creator name, but loses integrity and auditability.

#### source\_details

**What**: More detail about the origin, e.g., specific clinician or agency.  
**How is it designed**: Optional TEXT field.  
**How does it add value**: Helps verify or follow up on the alert source.  
**Questions to ask**: How much personal identifiable info should be allowed here?   
**What alternatives were considered**:

#### alert\_description

**What**: Case-specific notes for this alert instance.  
**How is it designed**: Optional free text field.  
**How does it add value**: Allows front-line staff to record additional context not captured in standardised fields.  
**Questions to ask**:

* How do we prevent sensitive or irrelevant info being entered here?
* Should there be flagging mechanisms to highlight alerts requiring privacy restrictions or special attention?

**What alternatives were considered**:

**Advisory Notes:**

* **Access Restrictions:** Because alert\_description can contain highly sensitive information (e.g., Domestic Abuse, Mental Health details), it is critical to restrict visibility based on user roles and responsibilities. For example, contractors performing repairs should **not** have access to detailed descriptions that could breach tenant privacy.
* **Audience Limitation:** System design should enforce audience restrictions at the application or database level to ensure that only authorized staff (e.g., housing officers, social workers, clinicians) can view sensitive notes.
* **Flagging:** There should be an explicit flag or indicator within the alert or notes to highlight when information requires additional confidentiality or specialized handling. This flag can trigger workflows or alerts for staff training and compliance.
* **Audit Trail:** All accesses and changes to this field should be logged to ensure accountability and traceability.

#### informed\_resident

**What**: Whether the tenant has been informed of the alert.  
**How is it designed**: Boolean with a default of FALSE.  
**How does it add value**: Helps ensure tenant communication and consent processes are followed.  
**Questions to ask**:

* Should this field be mandatory before an alert becomes active?
* Should date/time of informing also be captured?
* Should this be linked to notification table?

**What alternatives were considered**: Tracking full communication history in a separate table.

#### alert\_progress

**What**: Clinical or observational trend of the alert status.  
**How is it designed**: Coded VARCHAR(20) with a CHECK constraint for standard values: Stable, Improving, Deteriorating, Fluctuating, Unknown.  
**How does it add value**: Enables trend analysis and prioritisation over time. Standardizes tracking of condition trajectory for case management.  
**Questions to ask**:

* Should this be user-updated only, or could it derive from alert history?
* Are these categories sufficient or should more granular levels be added?

**What alternatives were considered**: Use a numeric scale or free text; less structured.

#### alert\_duration\_type

**What**: Expected length of time the alert is valid.  
**How is it designed**: Controlled list of values ('Long Term', 'Temporary', 'Unknown') using CHECK constraint.  
**How does it add value**: Supports planning for support interventions and reviews.  
**Questions to ask**:

* Should we allow custom duration categories?
* How to handle alerts that change duration type over time?

**What alternatives were considered**: Use a boolean flag like is\_temporary, but categorisation is more useful.

#### alert\_duration

**What**: Specific number of days the alert is expected to last.  
**How is it designed**: Optional integer field.  
**How does it add value**: Allows more precise tracking for temporary or scheduled alerts. Enables expiry calculations and reminders for review.  
**Questions to ask**: Should this be calculated based on start\_date and end\_date instead?  
**What alternatives were considered**: Omitting the field and using derived calculations. Calculate duration from dates, but stored value is simpler for queries.

#### start\_date

**What**: Date the alert starts or becomes active.  
**How is it designed**: Mandatory DATE field not null.  
**How does it add value**: Establishes the timeline of the alert.  
**Questions to ask**: Should this be editable after creation?  
**What alternatives were considered**: Automatically setting to CURRENT\_DATE, but explicit input is safer.

#### end\_date

**What**: Optional date for the alert to expire or be reviewed.  
**How is it designed**: Optional DATE field.  
**How does it add value**: Helps prevent stale alerts from remaining active indefinitely.  
**Questions to ask**: How to handle indefinite or open-ended alerts?  
**What alternatives were considered**:

#### review\_date

**What**: Date to review the alert for updates.  
**How is it designed**: Optional DATE field.  
**How does it add value**: Promotes regular review for continued relevance and accuracy.  
**Questions to ask**: Should this field be mandatory for some types of alerts?  
**What alternatives were considered**: Deriving review frequency from alert\_category\_types.

#### active

**What**: Whether the alert is currently in effect.  
**How is it designed**: Boolean, defaulting to TRUE.  
**How does it add value**: Allows alerts to remain in the record but inactive. Allows filtering of current versus historical alerts easily.  
**Questions to ask**: Should this be auto-set based on end\_date?  
**What alternatives were considered**: Using a computed status based on date comparisons.

**Advisory notes:**

* Systems should prevent "active = TRUE" if the end\_date is in the past, unless an override reason is provided.
* Optionally, create database constraints or triggers to auto-set active = FALSE when:
* end\_date is not null and less than today’s date.
* Or when a status field (if added later) changes to "Resolved" or "Closed".
* Staff responsible for managing or reviewing alerts (e.g., housing officers or clinicians) should have exclusive access to change this field.
* Systems should prevent non-authorized users (e.g., contractors) from reactivating or deactivating alerts manually.

#### last\_updated

**What**: Timestamp of the last update to this alert.  
**How is it designed**: Defaults to current timestamp on insert; updated on changes.  
**How does it add value**: Supports auditing, sorting, and syncing with other systems.  
**Questions to ask**: Should changes trigger automatic update timestamp?  
**What alternatives were considered**:

#### updated\_by\_user\_id

**What:** Reference to the user who last updated the alert record.

**How is it designed:** FK to person(person\_id). Optional but recommended for accountability.

**How does it add value:** Enhances accountability by identifying who made the most recent change. Enables filtered reporting, change tracking, and supports governance in regulated environments.

**Questions to ask:**

* Should updates by automated processes (e.g., system batch jobs) have a dedicated system user ID?
* Should we enforce updates to this field through application logic to ensure audit integrity?
* Should inactive or deleted users still be traceable in historic alert records?
* Should roles or permissions be enforced at the database level or in application logic?

**What alternatives were considered:** Free-text name.

## person\_alert\_history

### Purpose:

Tracks all meaningful changes made to person\_alert records for auditability, compliance, and accountability. This is especially important in housing and safeguarding scenarios, where decisions or oversights can have significant consequences for tenants or household members.

### Considerations:

* Enables a transparent audit trail of actions taken on alerts (e.g. who changed the status and how).
* Provides an immutable history of interactions for regulatory, legal, or QA review.
* Supports reporting on user behaviour, system changes, and alert lifecycle.

### Fields

#### person\_alert\_history\_id

**What:** Unique identifier for the history record.

**How is it designed:** INTEGER PRIMARY KEY AUTOINCREMENT

**How does it add value:** Ensures each change event is uniquely traceable.

**Questions to ask:**

**What alternatives were considered:**

#### person\_alert\_id

**What:** Foreign key to the alert record being changed.

**How is it designed:** INTEGER NOT NULL with FK to person\_alert(person\_alert\_id)

**How does it add value:** Links the change record to the alert it's describing.

**Questions to ask:** Should deletion of an alert be blocked if history exists?

**What alternatives were considered:** Storing alert\_code and person\_id redundantly (but those can be joined from the source).

#### action\_type

**What:** Indicates the kind of change.

**How is it designed:** VARCHAR(20) with a CHECK constraint for allowed values: INSERT, UPDATE, or DELETE.

**How does it add value:** Helps distinguish between creation, modification, and deletion events for auditing or rollback purposes.

**Questions to ask:** Are there more values to be added?

**Alternatives considered:** Look-up table – not necessary for just three values.

#### changed\_by\_user\_id

**What:** Person who performed the change.

**How is it designed:** INTEGER NOT NULL, FK to person(person\_id).

**How does it add value:** Supports audit trails and accountability.

**Questions to ask:** Should changes by system processes be logged under a special "System User"?

**What alternatives were considered:** Storing name or role (less reliable than linking to person table).

#### change\_timestamp

**What:** When the change occurred.

**How is it designed:** TIMESTAMP, defaults to CURRENT\_TIMESTAMP.

**How does it add value:** Provides chronological record of changes for traceability and event reconstruction.

**Questions to ask:** Should this be in UTC or localized?

**What alternatives were considered:** Manual timestamps from application (less consistent).

#### change\_summary

**What:** Description of what changed.

**How designed:** TEXT, optional.

**How it adds value:** Gives a human-readable explanation of the action taken, useful for auditors or reviewers.

**Questions to ask:** Should this follow a structured template (e.g. "Field X changed from Y to Z")?

**What alternatives were considered:**

# Look-up Tables

## person\_alert\_codes

### **Purpose:**

Stores a standardised reference list of alert codes representing medical or safeguarding conditions, vulnerabilities, and risk types that may impact housing services, repairs, or tenant wellbeing. It categorizes alerts into hierarchies (code/subcode) and aligns them to categories such as Medical, Risk, Safeguarding, etc.

This lookup table aims to enable consistency across data entry, analysis, integrations and regulatory compliance.

### Considerations

* This lookup table should be read-only in production and only editable by system administrators or data managers.
* Where possible, expose a dropdown UI based on alert\_name + alert\_subname.
* Consider adding date\_added, last\_reviewed, and status fields if future versioning or inactivation of codes is needed.
* Audit the criticality and Awaab flags periodically based on clinical guidance and evolving legislation.

### Fields

#### person\_alert\_code\_id

**What:** Unique internal identifier for each alert code entry. **How is it designed:** Auto-incrementing primary key integer. **How does it add value:** Provides a unique reference for joining data across tables while abstracting away from the public code structure. **Questions to ask:**

**What alternatives were considered:** Using alert\_code + alert\_subcode as a compound primary key, but a surrogate key simplifies references.

#### alert\_code

**What:** Top-level code representing a broad alert category (e.g., MED1 for Medical). **How is it designed:** Mandatory short string (VARCHAR(10)), manually defined. NOT NULL, enforced uniqueness alongside alert\_subcode. **How does it add value:** Allows grouping of subcodes into categories and helps with hierarchical filtering. Helps organize alerts into readable, thematic blocks, simplifies reporting and filtering.  
**Questions to ask:** Do we have a naming convention policy to avoid duplication or inconsistency? **What alternatives were considered:** Automatically generated codes, but manual input ensures semantic clarity and alignment with data standards.

#### alert\_subcode

**What:** The full code identifying a specific alert (e.g., MED1.01 for Asthma). **How is it designed:** Mandatory VARCHAR(15) field. NOT NULL, enforced uniqueness with alert\_code. **How does it add value:** Creates a readable and unique identifier for a specific condition or risk within a broader group.  
**Questions to ask:** Should alert\_subcode always be prefixed by alert\_code, or can it vary in structure?  
**What alternatives were considered:** Generating subcodes dynamically or using UUIDs, but structured codes are better for readability and filtering. Flat code system (e.g. MED001), less hierarchical.

#### alert\_category\_id

**What**: Links this alert code to a parent category (e.g. Medical, Behavioural, Safeguarding).

**How designed**: INTEGER NOT NULL FK to alert\_categories(alert\_category\_id).

**Value added:** Enables grouping, filtering, and category-specific logic or UI elements.

**Questions to ask:**

**Alternatives considered:** Storing category name directly (risks inconsistency).

#### alert\_name

**What:** Descriptive name for the alert group or condition type (e.g., “Respiratory Condition”). **How is it designed:** Required VARCHAR(100) NOT NULL. **How does it add value:** Helps staff understand the broad nature of the alert before reviewing details. **Questions to ask:**

* Do we want to align this with medical or social care ontologies?
* Should this be strictly controlled or open to editing?

**What alternatives were considered:** Using SNOMED or ICD-10 codes, but plain English was prioritised for housing context.

#### alert\_subname

**What:** Specific condition or risk within the alert group (e.g., “Asthma”). **How is it designed:** Required VARCHAR(100). **How does it add value:** Provides a clear and recognisable term for precise identification and communication.  
**Questions to ask:** Should we allow free text or pick from a standardised list (e.g., NHS condition codes)?  
**What alternatives were considered:** Mandatory selection from a controlled vocabulary, but plain English text allows for local flexibility.

#### is\_critical

**What**: Indicates whether the alert poses a critical risk to life or safety. **How is it designed:** Boolean field, defaulting to FALSE.  
**How does it add value:** Allows prioritisation in workflows and UI indicators for urgent cases.  
**Questions to ask:** Who determines criticality, the alert creator or a clinical authority?  
**What alternatives were considered:** Using a multi-level severity field instead of a binary flag.

#### is\_awaab\_relevant

**What:** Flags whether the alert is relevant under Awaab’s Law (e.g., linked to damp, mould, respiratory risk).  
**How is it designed:** Boolean field, defaulting to FALSE. **How does it add value**: Helps identify which alerts require enhanced responses under regulatory requirements.  
**Questions to ask:**

* Should this be dynamically derived from alert\_category or alert\_subname?
* Should this field evolve over time as legislation changes?

**What alternatives were considered:** A separate join table mapping alerts to hazard categories, but this flag is simpler for now.

#### repairs\_impact

**What:** Indicates how the alert affects or is affected by housing repairs. **How is it designed:** Controlled list (High, Medium, Low, None) with a CHECK constraint.  
**How does it add value:** Supports prioritised repairs scheduling and risk assessment during works. **Questions to ask:** Should this be enforced in job scheduling or just advisory?

**What alternatives were considered:** Numeric scoring, but a simple 4-tier model improves usability.

#### support\_required

**What:** Notes or instructions for specific support needs (e.g., “Use interpreter”, “Extra ventilation checks”).  
**How is it designed:** Free text field. **How does it add value:** Provides actionable instructions for contractors or frontline staff during interactions. **Questions to ask:   
What alternatives were considered:** Splitting into multiple binary or coded fields for common support types.

## source\_types

### Purpose

The source\_types table standardises the classification of sources from which a person alert originates. This includes professionals like GPs, social workers, and police, as well as self-reported alerts from tenants or family members. By assigning a code, label, and description to each source type, this lookup table enables consistent, reliable attribution of alerts, which is vital for auditing, triage, safeguarding decisions, and inter-agency communication.

It supports both analytical reporting (e.g., frequency of alerts from health services vs. tenant-reported issues) and operational logic (e.g., prioritising alerts from verified medical professionals).

### Considerations

* **Consistency Across Records**: Centralising the list avoids the variability of free-text source inputs like "gp", "GP", "doctor", or "family".
* **Audit & Traceability**: Knowing who reported an issue is key for understanding urgency, credibility, and the appropriate response team.
* **User Interface Friendliness**: The short source\_code can be used for internal referencing or exports, while the source\_label provides readable context in UI and reports.
* **Data Governance**: A controlled list prevents entry errors and enables standardised filtering, grouping, and validation across the system.
* **Extensibility**: New types (e.g., school counsellors, fire services) can be added without altering the schema.
* **Privacy and Role-Based Access**: Some source types (e.g., police, GP) may imply sensitive content, this can influence how alerts are handled or restricted in the UI.

### Fields

#### source\_type\_id

**What:** Unique internal identifier for each source type.  
**How is it designed:** INTEGER PRIMARY KEY AUTOINCREMENT.  
**How does it add value:** Ensures a stable surrogate key for joins and references (e.g. in person\_alert).  
**Questions to ask:**   
**What alternatives were considered:** Using source\_code as the primary key — less flexible for indexing or referencing across multiple systems.

#### source\_code

**What:** Short alphanumeric code to identify the source type (e.g., GP, SELF, POL).  
**How is it designed:** VARCHAR(10) NOT NULL UNIQUE.  
**How does it add value:** Human-readable and easily referenced in UI, exports, or integrations.  
**Questions to ask:** Should this follow a strict naming convention across domains?  
**What alternatives were considered:** Using numeric codes, more compact but less intuitive.

#### source\_name

**What:** Full descriptive name of the source type (e.g., “General Practitioner”).  
**How is it designed:** VARCHAR(100) NOT NULL.  
**How does it add value:** Supports clear display in user interfaces and reports.  
**Questions to ask:**   
**What alternatives were considered:**

#### description

**What:** Optional notes to provide context or clarification for the source type.  
**How is it designed:** TEXT, nullable.  
**How does it add value:** Improves clarity for administrators or developers maintaining the list; useful in admin UIs or documentation.  
**Questions to ask:** Should this include guidance for when to use this source versus others?  
**What alternatives were considered:**

## alert\_categories

### Purpose

The alert\_categories table defines the high-level classifications of alerts, such as Medical, Mental Health, Risk, or Safeguarding. These categories are designed to group related alert codes into broader themes for reporting, filtering, role-based access control, and operational workflows.

This lookup table supports standardisation across the person\_alert\_codes table (via the alert\_category\_id foreign key), ensuring consistent categorisation of tenant risks or support needs. It also supports integration across modules like repairs, tenancy, or health and safety.

### Considerations

* **Data Normalisation**: Prevents inconsistent manual entry of categories like "medical", "MED", or "med condition".
* **UI & UX Benefits**: Allows dropdowns or filtering tools in the interface to use friendly names with consistent IDs.
* **Governance**: Aids in future data governance and analytics by encouraging consistency across systems.
* **Access Control**: May be used in the application layer to restrict access to sensitive categories (e.g. Safeguarding).
* **Extensibility**: New categories can be added through data governance processes without schema changes.
* **Interoperability**: Supports mapping to national or organisational data standards (e.g., HACT, MHCLG).
* **Fallback Options**: Includes an "Unknown" category to handle legacy or incomplete data.

### Fields

#### alert\_category\_id

**What:** Unique internal ID for each alert category.

**How is it designed:** Auto-incrementing integer, serves as the primary key.

**How does it add value:** Supports foreign key relationships to other tables (e.g. person\_alert\_codes) for consistency and performance.

**Questions to ask:** Should these IDs ever be exposed to users, or remain hidden behind friendly names?

**What alternatives were considered:** Using the name itself as the key (e.g. ENUM): discarded to maintain flexibility and avoid issues with renaming.

#### alert\_category\_name

**What:** Human-readable name of the category (e.g. "Mental Health", "Safeguarding").

**How is it designed:** VARCHAR(50), enforced as UNIQUE NOT NULL.

**How does it add value:** Ensures clarity and usability in dropdowns, filters, reports, and UI.

**Questions to ask:** Should descriptions also be included?

**What alternatives were considered:**

* Using abbreviations or codes instead of full labels - rejected in favour of clarity.
* ENUMs - not chosen to allow easier updates without schema changes.