Person Alert Code

**Design Decisions**

*Proposed Sector-Wide Standard for Housing Providers (HACT/MHCLG aligned)*  
**Version:** Draft v1.0 | **Prepared by:** Elena Iurco | **Date:** 25 July 2025

# Table: person\_alert

## person\_alert\_id

**What**: Unique identifier for each alert instance.  
**How is it designed**: An auto-incrementing integer primary key.  
**How does it add value**: Ensures each alert is uniquely identifiable for database integrity and traceability.  
**Questions to ask**:   
**What alternatives were considered**: None — this is fundamental for the table's function.

## person\_id

**What**: Reference to the person the alert applies to.  
**How is it designed**: A mandatory foreign key (not null) linking to the person table.  
**How does it add value**: Ties alerts directly to individuals for accurate data management.  
**Questions to ask**:  
**What alternatives were considered**: None — this is fundamental for the table's function.

## alert\_code\_id

**What**: Reference to the standardised alert code from person\_alert\_codes.  
**How is it designed**: A mandatory foreign key.  
**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?  
**What alternatives were considered**: Embedding codes directly as text, but that would undermine data consistency.

## 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?  
**What alternatives were considered**:

## alert\_source

**What**: The origin of the alert information (e.g. NHS, Self-reported).  
**How is it designed**: Optional VARCHAR(100).  
**How does it add value**: Provides transparency and credibility of the alert; informs how seriously to treat it.  
**Questions to ask**: Should this be a dropdown or code list instead of free text?  
**What alternatives were considered**:

## created\_by

**What**: Name of the person who created the alert.  
**How is it designed**: Optional VARCHAR(100).  
**How does it add value**: Supports audit trail and accountability.  
**Questions to ask**: Do we want to log user IDs instead of names for traceability?  
**What alternatives were considered**: Capturing both name and internal user ID.

## source\_description

**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**:

## 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?  
**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.  
**Questions to ask**: Should this be user-updated only, or could it derive from alert history?  
**What alternatives were considered**:

## 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?  
**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.  
**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.

## start\_date

**What**: Date the alert starts or becomes active.  
**How is it designed**: Mandatory DATE field.  
**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**:  
**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.

## 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.  
**Questions to ask**: Should this be auto-set based on end\_date?  
**What alternatives were considered**: Using a computed status based on date comparisons.

## last\_updated

**What**: Timestamp of the last update to this alert.  
**How is it designed**: Defaults to CURRENT\_TIMESTAMP.  
**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**: Adding a full update history table.

## updated\_by

**What**: The person who last updated the alert.  
**How is it designed**: Optional VARCHAR(100).  
**How does it add value**: Ensures accountability and traceability of changes.  
**Questions to ask**: Should this store a unique user ID instead of a name string?  
**What alternatives were considered**: Capturing user IDs from an internal staff directory.

# Table: person\_alert\_codes

## 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:** Do we need to expose this ID to APIs or interfaces, or keep it purely internal? **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: Respiratory). **How is it designed:** Mandatory short string (VARCHAR(10)), manually defined. **How does it add value:** Allows grouping of subcodes into categories and helps with hierarchical 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. **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.

## alert\_category

**What:** The general category of alert, e.g., Medical, Risk, Environmental. **How is it designed:** Free text field (VARCHAR(50)). **How does it add value:** Enables high-level grouping and analytics on alert types across departments or systems.  
**Questions to ask:** Should we use a controlled list here instead of free text to ensure consistency?  
**What alternatives were considered:** Using ENUM or a lookup table — still under consideration for future data governance improvements.

## alert\_name

**What:** Descriptive name for the alert group or condition type (e.g., “Respiratory Condition”). **How is it designed:** Required VARCHAR(100). **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? **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 operational response, including escalation and compliance measures.  
**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?  
**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 repairs impact be derived from property condition or manually input per alert? **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.