ChatGPT

SensibLaw Ontology & Database Design Specification

Overview: Three-Layer Ontology Structure

SensibLaw's data model is built on **three conceptual layers** that separate legal contexts, abstract wrongs, and concrete events [1]. This layered ontology ensures clarity and flexibility across different legal traditions and real-world scenarios:

- **Layer 1 – Normative Systems & Sources:** Captures the context of laws (e.g. jurisdictions, legal traditions, sources of norms). This is *where* rules live – such as a nation's common law, an indigenous legal tradition, a religious legal system, etc [2].
- **Layer 2 – Wrong Types (Abstract Wrongs):** Represents generalized patterns of wrongs or liabilities in a given system. These are akin to torts, delicts, tikanga wrongs, human rights violations, etc. – the *conceptual wrong* as defined by a legal system [3] [4].
- **Layer 3 – Events & Harms:** Represents specific real-world incidents and their consequences. These are the *instances* (e.g. an actual event causing harm to a person or community) that may be analyzed under one or more wrong types [5].

Everything in SensibLaw's schema attaches to one of these layers. By separating them, we avoid mixing abstract legal definitions with factual events, enabling the system to support cross-jurisdictional and cross-cultural comparisons without hardcoding one legal perspective [6] [7].

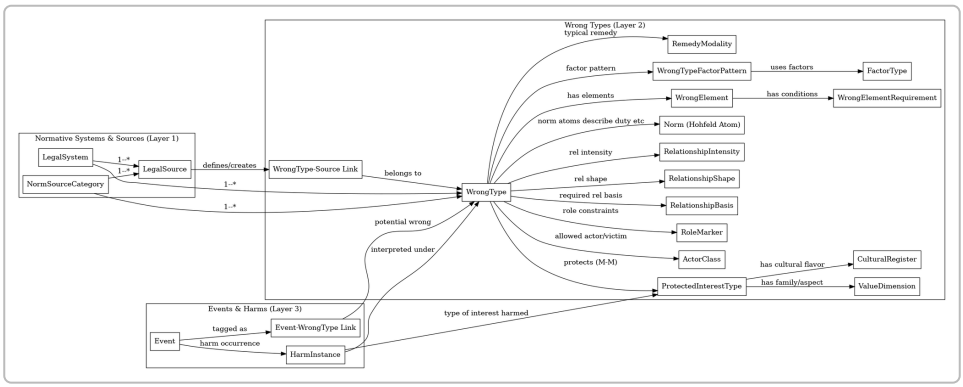

Figure: Entity-Relationship overview of SensibLaw's ontology (grouped by the three layers). Core entities and their relationships are shown. For clarity, arrow labels indicate the nature of relationships (e.g. WrongType "protects" a ProtectedInterestType, Event "tagged as" a WrongType). Primary keys are internal IDs (not shown), with foreign key links between tables as arrows.

Layer 1: Normative Systems & Sources (Legal Context)

Layer 1 defines *legal system metadata* – essentially, **where a rule comes from** and what kind of source it is:

- **LegalSystem:** Each entry represents a distinct legal system or context. This could be a national legal system or a cultural/legal tradition. For example: "Australian common law", "New Zealand Tikanga (Māori customary law)", "Pakistani Islamic (Hanafi) law", or "Chinese Civil Code system" [8]. Key attributes might include:
- *Jurisdiction or community* (nation, tribe, religion, etc.),
- *Tradition* (common law, civil law, Indigenous, religious, etc.),

- *Source form* (whether its rules come mainly from statutes, case law, custom, religious texts, etc.). Each LegalSystem has a unique internal ID and possibly a human-readable code (like `AU.COMMON` for Australian common law), but that code is used only as a label – all relationships use the internal ID to avoid coupling logic to strings [9]. This table is referenced by many others to situate every law or wrong in context [10].

- **NormSourceCategory:** This is an enumeration of *types of legal sources* [11]. It classifies the form of authority for a legal norm. Examples might include:

- **Statute** (legislation/acts),
- **CaseLaw** (judicial decisions),
- **Constitution**,
- **Treaty/International Law**,
- **Customary** (customary law or Indigenous authorities),

- **ReligiousText** (e.g. Quran, Bible verses), etc.
  A WrongType will often cite a primary source category (e.g. negligence in Australia is rooted in case law, whereas a Māori wrong might be rooted in customary norms) [12]. NormSourceCategory entries have an ID and a label (e.g. "statute", "case law"), and are referenced by both **LegalSource** and **WrongType** to describe the nature of their authority [13].

- **LegalSource:** Each record corresponds to a specific legal document or source **within a LegalSystem** [14]. This can be:

- A specific Act or statute section (e.g. *Torts Act 1881 §5*),
- A judicial case (e.g. *Donoghue v Stevenson [1932]*),
- A constitutional article, treaty provision, a tikanga statement or proverb, a religious verse, etc.
  Fields include a unique ID, a reference to its **LegalSystem**, a reference to its **NormSourceCategory**, and citation details (names, section numbers, court references, etc.) [15]. By linking to LegalSystem, we can differentiate sources even if they have similar names (e.g. distinguish a section "5" of an Australian Act vs. a Canadian Act).

Relationships and Keys: A LegalSource points to the LegalSystem it belongs to and its source category via foreign keys [9]. In turn, higher-layer concepts will link back to these sources. For example, a WrongType may be defined or shaped by one or more LegalSource entries (see "WrongType–Source Link" below). All cross-references use internal IDs, ensuring we never rely on text labels or names for linkage (this avoids issues like name changes or duplicates) [10].

Layer 2: Wrong Types (Abstract Wrongful Acts)

Layer 2 captures the heart of the ontology: the **abstract "wrongs"** recognized by various legal systems. This is analogous to causes of action or offense types, but generalized to accommodate *any* normative system (tort, delict, indigenous harm, etc.) without bias [16] [17]. Key structures include:

- **WrongType:** This is the central entity representing a pattern of actionable wrong in a specific system [4]. It defines *what kind of misdeed or harm* we're dealing with. Each WrongType has:
- **ID** (primary key) and an optional human-readable code (e.g. `"AU.NEG.PHYS_INJ"` for *Australian Negligence – Physical Injury*).
- **LegalSystem ID** (foreign key): which legal system defines this wrong [18]. For instance, a WrongType for "defamation" might exist separately for Australia vs. for an Islamic context, each linked to the respective system.
- **NormSourceCategory ID** (foreign key): what type of authority primarily defines this wrong [12] (statutory tort vs. common law tort, vs. customary norm, etc.).

- *Other descriptive fields:* possibly a name and description of the wrong. Examples of WrongType instances: *"Negligence (physical injury)" in Australian common law*, *"Defamation" in NZ common law*, *"Harm to mana" in Māori tikanga*, *"Qadhf (false accusation)" in Pakistani Islamic law*, *"Privacy breach (personal data)" under Chinese Civil Code* [19]. Despite different cultural contexts, these all share the same structural schema in WrongType, illustrating the flexible, comparative design [16] [17].

- **WrongType–Source Link:** This is a join table (or associative entity) linking **WrongType** to one or more **LegalSource** records [20]. It captures the relationship between an abstract wrong and the legal texts or precedents that define it. For example, one entry might link the *"Negligence" WrongType* to *Donoghue v Stevenson (a leading case)*, marked with a relation type "leading_case". Another entry might link *"Negligence"* to a section of a Civil Liability Act with relation "creates" (indicating that statute defines or creates the cause of action) [21]. Each link could carry a attribute like *relation type* (e.g. *creates, defines, modifies, cites, leading_case*) to clarify how the source is related [21]. In terms of keys: this table would have its own ID and foreign keys referencing a WrongType and a LegalSource (forming a many-to-many relationship between them). This design lets us flexibly attach any number of statutes or cases to a WrongType without duplicating the WrongType or conflating distinct legal sources.

Linking Laws to Wrongs: Through WrongType and its source links, the model allows robust linking to **statutes, case law, and even non-Western sources**. For instance, a WrongType in an Indigenous context can link to a *tikanga* **LegalSource** (like a written record of a customary principle or a well-known proverb) with a category "customary" [2] [11]. Similarly, a WrongType for a religious wrong could link to verses in a holy text. This *flexible linking* ensures that laws and norms from any system can be attached to the wrong's definition, rather than hardcoding only statutes or cases.

Protected Interests & Values

A core question for any wrong is: **What interest or value does it protect?** SensibLaw addresses this by modeling protected interests explicitly, rather than implicitly in the wrong's name. The design breaks down interests into atomic components to allow multi-faceted harms and avoid an explosion of hardcoded categories [22] [23]:

- **ValueDimension:** An abstract category of value that may be protected. We use a two-part taxonomy: a **family** and an **aspect** [24]. The "family" is a broad type of value (e.g. *integrity*, *status*,

control, *use*, *autonomy*, *development*), and the "aspect" specifies the context (e.g. *body*, *reputation*, *information*, *territory*, *child*) [25]. For example:

- `integrity.body` (physical integrity),
- `status.reputation` (reputational status),
- `control.information` (control over personal information).

This two-tier approach keeps the vocabulary consistent yet extensible (new aspects can be added under a finite set of families).

- **CulturalRegister:** A culturally specific expression or "flavor" of a value dimension [26]. Different cultures or legal systems often have unique concepts for similar underlying values. For instance:

- *mana* (a Māori concept of prestige, authority, and spiritual power) relates to status/honour,
- *face* (East Asian concept of dignity/honour) relates to reputation,
- *izzat* (South Asian concept of honour) also relates to status.

The CulturalRegister table lets us tag a value with one of these cultural nuances when relevant [27]. It's optional – a wrong can protect a value in a general sense (no specific cultural register, e.g. privacy in a generic sense) or within a cultural concept (e.g. a Māori wrong might specifically protect "status.reputation with cultural register = mana" [27]). This design prevents conflating culturally specific harms into entirely separate structural categories – instead, they're variations on the same underlying value theme.

- **ProtectedInterestType:** This table combines the above into a named interest that a wrong can protect [28]. Think of it as a *type of interest* in the abstract, not tied to any one person or case. Examples:

- "Physical integrity of persons" (could be defined by `integrity.body` + no cultural register, representing bodily safety),
- "Ecological integrity of land/Country" (maybe `integrity.ecological` + possibly a register if a specific culture's concept of land stewardship is invoked),
- "Community mana" (`status.reputation` + cultural register *mana*),
- "Privacy of personal information" (`control.information` with no specific register).

A ProtectedInterestType record might have fields: ID, a name/description, a foreign key to **ValueDimension** and optionally to **CulturalRegister** (if this interest involves a specific cultural concept) [29]. Importantly, these interest types do **not** include who holds the interest – they are generic. *Who* is affected will come into play at the event layer. This separation means we don't need separate wrong categories for each possible victim or group; the wrong focuses on *what kind of harm* it addresses [29].

- **Linking WrongTypes to Interests:** A WrongType can be linked to one or more ProtectedInterestType entries (many-to-many relationship) to declare **which interests that wrong is meant to protect** [30]. For example, a defamation WrongType might link to "status.reputation" interest; a negligence WrongType might link to "integrity.body" (physical safety) and also "integrity.property" if it covers property damage; an Indigenous sacred-site desecration WrongType might link to "integrity.ecological" and "status.spiritual" (if we had such a dimension) with a cultural register of that community's concept of sacredness. These links are *associative records* (e.g. a table WrongType_ProtectedInterest) containing WrongType ID and ProtectedInterestType ID [30].

By decoupling interests from wrong types, the schema avoids the pitfall of proliferating categories (e.g. having separate wrong types for "negligence causing physical injury" vs "negligence causing economic loss") [22] [23]. Instead, a single WrongType "Negligence" can link to multiple interest types (physical integrity, economic interests, etc.) and specific **WrongElementRequirements** (discussed later) can further condition the application. It also allows **multi-interest harms** – an event can implicate multiple interests (say, an incident harms both a person's body and dignity) without needing to treat it as two separate wrongs [31]. The ProtectedInterestType provides a common reference point, so even if two legal systems label things differently, we can recognize overlap (e.g. Western privacy vs. Indigenous concept of sacred knowledge might both map to a `control.information` value, aiding comparability).

Actors, Roles, and Relationships

Legal wrongs often involve specific types of actors (e.g. minors, government bodies) and relationships (e.g. employer-employee, caregiver-ward). To model this without hardcoding one society's roles, SensibLaw breaks down the participant schema into **modular components** [32] [33]:

- **ActorClass:** A coarse classification of an actor (participant) in an event [34]. This abstracts the **legal or ontological status** of actors. Examples:
- Human individual,
- Legal Person (e.g. corporation),
- State or Government body,
- Indigenous Collective or Tribe,

- Environment as a legal person (e.g. a river recognized as a legal entity).
  ActorClass answers "what kind of entity is this party?" This is important because some wrongs can only be committed by certain actor classes (for instance, only a *state actor* can violate certain human rights, or only a *person* can commit a personal tort).

- **RoleMarker:** A contextual or social role that an actor holds in the scenario [35]. These are like tags that mark someone as being in a special capacity relative to another. Examples:

- Child, Guardian, Parent,
- Employer, Employee,
- Elder, CommunityLeader,

- Official, Prisoner, **etc.**
  RoleMarkers are used to refine actor roles beyond the broad class. For example, an **ActorClass** might be "human" but with a RoleMarker "child" vs. "guardian". RoleMarkers often come in complementary pairs or sets (child/guardian, employer/employee) that might figure into relationship definitions.

- **RelationshipBasis:** The fundamental basis of a relationship between two parties [36]. This describes *why or how two entities are connected*. Examples of relationship bases:

- Kinship (family relationships),
- Employment (one hires the other),
- Custody/Guardianship (one is responsible for care of the other),
- Contractual (bound by a contract),
- Community membership,

- Authority (one has formal authority over the other, like state over citizen), etc.
  The **basis** is separated from the actors themselves; it's an abstract connection type.

- **RelationshipShape** & **RelationshipIntensity:** These qualify the nature of the relationship's structure and strength [37].

- *Shape* captures the power or hierarchy structure: e.g. **vertical** (one party has power over the other, like parent-child or employer-employee), **horizontal** (peers), **mutual** (e.g. siblings in some contexts), or **collective** (many-to-many relationships in a group) [37].
- *Intensity* captures how strong or enduring the relationship bond is: **strong** (deep, ongoing obligations), **weak** (minimal or occasional connection), or **incidental** (contextual/one-off) [37].
  For instance, a parent-child kinship would typically be "vertical + strong", whereas co-workers might be "horizontal + weak" (peers with a loose bond).

These building blocks can be combined to describe complex role relationships **without creating a single rigid enumeration** [38]. For example, what earlier might have been called a monolithic category "KINSHIP_GUARDIANSHIP" is now represented as:
- **RelationshipBasis** = kinship,
- a RoleMarker on one side = "guardian" (caregiver role), another RoleMarker = perhaps "child" on the other side,
- **Shape** = vertical, **Intensity** = strong (implying a clear hierarchy and durable bond) [39] [40].

WrongType Party Constraints: For each WrongType, the schema allows specifying constraints on the parties involved [41]: - Who can be the "perpetrator" or liable actor (e.g. must be an **ActorClass** = "Legal Person (company)" for certain regulatory wrongs)? And optionally, what RoleMarkers must they have (e.g. "employer")? [42]
- Who can be the "victim" or harmed party (e.g. **ActorClass** = "Human" with RoleMarker "employee")? [43]
These constraints are stored by referencing the relevant ActorClass and RoleMarker IDs. For example, a WrongType for *workplace harassment* might constrain: **ActorClass** of defendant = legal person or human (either a boss or the employing company), with RoleMarker "employer"; and plaintiff = human with RoleMarker "employee". Another WrongType like *Indigenous elder abuse* might specify the victim must have RoleMarker "elder" within a kinship basis.

WrongType Relationship Constraints: In addition, a WrongType can require that the two parties stand in a certain **relationship** to each other. This is modeled by referencing RelationshipBasis (and possibly Shape/Intensity) records. For example: - **Vicarious liability** WrongType (an employer held liable for employee's act) would include a constraint: RelationshipBasis = employment, Shape = vertical, Intensity = strong [40]. This encodes "the parties must be in a strong, hierarchical employment relationship".
- An Indigenous WrongType might require **kinship + care** basis with strong intensity (e.g. a guardian-relative caring for a child, as in a customary duty situation) [40].
- **State duty** WrongTypes (like human rights violations) might require basis = authority or custody (state vs detainee), shape = vertical, intensity = strong [40].

All these constraints are linked via foreign keys: WrongType has associated IDs or through join tables to the allowed ActorClass/RoleMarker for each role, and required RelationshipBasis/Shape/Intensity (if any) [44]. Internally, this could be implemented with tables like WrongType_AllowedActor (for each role in the norm, linking WrongType to ActorClass/RoleMarker for "actor" vs "victim" roles), and WrongType_RelationshipReq (linking to a RelationshipBasis record plus flags for required shape/intensity). The key point is that we **don't hardcode strings like "KINSHIP_CARE_VERTICAL" in the wrong type** – instead we store the components, which avoids a combinatorial explosion of categories

[45] [46]. The system can then evaluate, for a given event, whether the parties meet the required relationship conditions by checking these components.

This granular approach to actors and relationships addresses a major modeling pitfall: previously, one might be tempted to create a special-case category for each scenario (e.g. "Parent-child negligence" vs "Stranger negligence"), or to assume all torts are between similar actors. SensibLaw's ontology avoids that by letting us mix and match actor constraints and relationship requirements as needed [47] [38]. It's expressive enough to capture nuances across Western and Indigenous contexts without collapsing them into one-off exceptions. For instance, a tikanga-based wrong might simply use a different combination of ActorClass (e.g. "Indigenous collective" as victim), RoleMarkers ("elder"), and RelationshipBasis ("community/kinship") but **within the same schema** as any other wrong [16] [17].

Norms and Obligations (Hohfeldian Atoms)

To represent the **normative logic** of laws (duties, rights, permissions, etc.), SensibLaw uses an approach inspired by *Hohfeld's framework* of fundamental legal relations. Instead of storing a whole legal rule as free text, the system breaks each rule into **atomic normative statements** that can be reconstructed into a logic tree [48] [49]. This is essentially the **logical form of the law**:

- **Norm (Normative Atom):** Each WrongType is associated with one or more Norm atoms that describe the *underlying legal norms* it entails [50]. A Norm record typically includes:
- Reference to the **WrongType** it belongs to (linking it to context like actors, interests, etc.).
- A **Hohfeldian modality**: e.g. Duty, Right, Privilege, No-Right, Power, Liability, Immunity, etc. (For simplicity, many wrongs focus on duties and correlative rights: e.g. "X has a duty not to do Y" corresponds to "Y has a right not to have X do that").
- **Subject Role** and **Object Role**: placeholders tying the norm to the WrongType's party roles. For example, a Norm might be "*Defendant* (subject) **must not** do [action] to *Plaintiff* (object)…". The actual labels like "defendant" or "plaintiff" or "child" can be drawn from the WrongType's defined roles [51]. Essentially, the norm uses role markers like variables.
- **Action kind:** what action or omission the norm covers (e.g. *do* something, *omit* something, *cause harm*, *disclose information*, *enter land*, *speak defamatory words*, etc.) [52].
- **Object kind:** the kind of object/interest the action is applied to (e.g. an object could be a *physical body*, a *piece of property*, a *reputation*, etc., depending on what the wrong concerns) [53].
- **Context/Condition** (if any): conditions under which the norm applies or exceptions. For instance, a norm might only apply "in the context of an employment relationship" or "unless consent is given" – these can reference RelationshipBasis or other factors as needed [54]. Complex conditions can also be encoded as separate Norm atoms (e.g. an exception might be another Norm that modulates the primary one).

Using these Norm entries, we can reconstruct a full legal rule logically. For example, consider a rule: "A person must not sell spray paint unless they have a retailer license." This might be modeled with a Norm: Subject = Actor ("person"), Action = "sell spray paint", Object = (implicitly the product, spray paint), Modality = Duty (must not), and a condition "unless (subject) has license" which could be represented as an Exception norm or a conditional clause. In SensibLaw's terms, that would be broken into nodes of a logic tree [55] [56]. Each clause (obligation, exception, condition) becomes an explicit node in a graph structure, traceable back to the text [56] [57].

From a database perspective, Norm atoms might be stored in a table with columns for WrongType ID, modality (duty/right/etc), subject role label, object role label, action reference, object kind reference, and perhaps a pointer to a condition expression or linked factor. Because these references tie into the controlled vocabularies (roles, action types, object types, relationship contexts), the norms are **fully**

machine-readable and composable. This approach ensures the **implicit logic of a law is made explicit** in the data [56] [57]. It also means we are not locked to any one language – norms from an English statute or a Māori tikanga can both be mapped into this structured form, enabling cross-jurisdictional reasoning (the system can map different terminologies to a **common ontology of concepts** behind the scenes [58] [59]).

In summary, Norm atoms link the abstract WrongType to concrete legal obligations or permissions, using consistent building blocks. They answer "What *must* or *must not* happen, and who holds those duties/rights?" [51]. By storing norms in atomic form, SensibLaw can generate **logic trees** for legal reasoning – for instance, building a proof tree to show how a conclusion (liability) is reached through these normative clauses [60] [61]. This is the basis of SensibLaw's reasoning engine that can trace step-by-step which laws and conditions apply in a scenario [60] [62].

Wrong Elements and Factor Patterns

Many legal wrongs (especially torts and delicts) are analyzed by breaking them into **elements** (like elements of a cause of action) and considering various **factors** that influence outcomes. The ontology includes tables to represent these doctrinal details, which help in reasoning and comparing how decisions are made:

- **WrongElement:** This represents a named component of the WrongType's legal test [63]. For example, a negligence WrongType might have elements: "duty of care", "breach of duty", "causation", "damage". A defamation WrongType might have elements like "publication", "identification (of plaintiff)", "defamatory meaning", etc. Each WrongElement is linked to a specific WrongType (foreign key) and has a name/type. These essentially enumerate the checklist a court goes through for that wrong.

- **WrongElementRequirement:** This table attaches specific **conditions or sub-requirements** to a WrongElement [64]. It allows encoding special rules or thresholds. For instance:

- For the "damage" element in negligence, a WrongElementRequirement might specify "HarmType must be physical injury or property damage; if purely economic loss, then additional criteria X must be met" [64].

- Or for a "status" element in an Indigenous wrong, a requirement might be "the victim must have status as an elder in the community" (linking back to a RoleMarker perhaps). Each requirement could reference other parts of the ontology: e.g. a requirement might say "harm's ValueDimension = integrity.body" (meaning the harm must be bodily) [65]. These act as filters or prerequisites that need to be satisfied for that element to be fulfilled.

- **FactorType:** A library of common evaluative **factors** used in legal reasoning [66]. These factors are typically boolean or weighted considerations that can tilt a decision one way or another. For example, in negligence or human-rights cases:

- "Defendant knew of the risk" (knowledge-based factor),
- "Plaintiff was especially vulnerable" (situation-based factor),
- "Defendant had a special relationship to plaintiff" (relationship-based factor),
- "Social utility of defendant's conduct" (policy-based factor),

- etc.
  Each FactorType has an ID, a short description, and possibly a category (like we might categorize

factors as **policy** factors, **moral** factors, **relationship** factors, etc., for organizational purposes [67]). These factors are **reusable across WrongTypes** – they are not specific to one jurisdiction. (This aligns with AI approaches like Bench-Capon's factor-based reasoning: abstract factors that appear across many cases [67].)

- **WrongTypeFactorPattern:** This ties specific FactorTypes to a WrongType, indicating which factors are relevant to that wrong and how they affect the outcome [68]. For example, for negligence, relevant factors might be:

- "plaintiff vulnerable" (supports finding a duty of care if true),
- "risk obvious" (might support *no* liability if true, as plaintiff could have avoided it, etc.),
- "countervailing social utility" (supports no liability),
- "defendant's knowledge of risk" (supports liability).
  For each FactorType attached, the pattern would denote whether it *tends to support liability* or *no liability* (or perhaps neutral) [68]. Essentially, this is a template of argument: when analyzing a case under this WrongType, these are the points to consider. If using a graph database or rule engine, these factors could even be connected to case outcomes to see how they were applied historically. They provide an **explanation layer** – why did a case go one way? Because X, Y, Z factors favored the plaintiff, etc.

These elements and factors aren't foreign keys in a strict relational sense across all tables (they don't all join to other main tables except linking to WrongType or to ValueDimension for requirements), but they are crucial for the *semantic completeness* of the ontology [69]. They allow developers to capture detailed legal logic without losing the big connections: - **Elements** structure the wrong into parts (making it easier to implement forms or checklists in an application – e.g., "Did the plaintiff prove element A? Element B?"). - **Requirements** on elements add nuance and ensure we can model specialized doctrines (like the difference between physical harm vs economic loss in negligence) without multiplying WrongTypes [70]. - **Factors** and **patterns** provide a bridge to AI and case-law reasoning, enabling comparisons across jurisdictions (e.g., both UK and US negligence cases consider "obvious risk" – we can tag that same FactorType in both and compare outcomes).

Remedies and Outcomes

The model also accounts for the **outcomes or remedies** associated with wrongs. Instead of treating remedies from Western law (damages, injunctions) as fundamentally different from those in Indigenous or other systems (restoration ceremonies, apologies), SensibLaw abstracts remedies as **general modalities** [71]. Each WrongType can then link to the remedies typically available for it [72]:

- **RemedyModality:** A list of possible remedy types, defined broadly. Examples of modalities (drawn from the design):
- **Monetary Compensation** (substitutive money damages),
- **Restoration in Kind** (repair or restore what was harmed, e.g. fix property, environmental remediation),
- **Injunction/Order** (court order to stop or do something, e.g. "stop the harmful activity"),
- **Declaration** (official statement of rights/status, e.g. declaratory judgment),
- **Relationship Repair Process** (like mediation, peacemaking processes, restorative justice meetings),
- **Status Restoration** (restoring the victim's honor or mana or rights, which could include ceremonies or public statements to restore reputation [73]),
- **Public Apology**,

- **Collective Reparation** (community-level remedies, could involve community service or group compensation),
- **Ritual / Ceremonial Act** (e.g. a specific ritual to heal or apologize, common in some Indigenous contexts).
  This list is extensible but is meant to cover both common law remedies and those from other traditions in one taxonomy [74].

Each WrongType can be associated with one or more RemedyModalities (many-to-many link) indicating which remedies are applicable [75]. For example, a WrongType for defamation might allow *damages, apology, injunction*; a Māori wrong involving mana might allow *relationship repair process, status restoration ceremony, apology*, etc. By normalizing remedies this way, we **avoid treating Indigenous or non-Western remedies as "other"** – they simply appear as different combinations of the same universal remedy types [75]. This is both a design philosophy (to treat all systems with parity) and a practical benefit (developers can implement remedy logic generically, e.g. a UI that lists possible remedies for a wrong can populate from this table, whether it's a common law tort or a tikanga harm).

Layer 3: Events & Harm Instances (TiRCorder Integration)

The third layer handles **concrete events** – things that happen in the real world – and ties them back into the legal ontology. In the context of the ITIR suite, events are often captured via **TiRCorder**, a trauma-informed recorder tool (e.g., voice-recorded narratives of incidents) [76] [77]. SensibLaw's schema provides a way to store these narratives as structured data and link them to the legal concepts above:

- **Event:** A record of a specific occurrence in time involving one or more actors, which potentially caused harm [78]. An Event would have attributes like:
- **Timestamp/Timeframe** (when it occurred),
- **Location** (could be a place or jurisdiction),
- **Description** (text or references to transcripts, etc. from TiRCorder),

- Possibly a unique ID or reference to a TiRCorder recording.
  Crucially, an Event will have relationships to **participants** – e.g., link to the actual people or entities involved. In a fully fleshed design, we might have an EventParticipant table linking an Event to an **Entity** (like a person record) plus a role (perhaps tying back to RoleMarkers like "this participant was in role X for this event"). However, to keep the ontology conceptual, the primary links emphasized are those to wrong types and harms (the assumption is that the identities of actors would be handled by a separate module, or the Event might reference them via an external ID or narrative text).

- **Event–WrongType Link:** An association table linking an Event to one or more WrongType(s) [79]. This denotes that *"this event might instantiate or relate to that abstract wrong."* In practice, an event could give rise to multiple legal analyses – for example, a single incident (a protest turned violent) might be tagged as potentially **"assault"** and **"human rights violation"** and **"breach of peace (customary law)"**, etc. Each link could also store metadata like a confidence score (if auto-classified) or a status (e.g. confirmed vs. just suggested) [80]. By tagging events with WrongTypes, the system enables searching or reasoning like "find all events in the dataset that look like defamation cases" or "show me events of type 'harm to mana'". This is the bridge from raw narratives to legal categories.

- **HarmInstance:** This table captures the specific harms suffered in an Event, on a per-affected-party basis [81]. Each HarmInstance answers: *who or what was harmed, and in what way?* Key fields:

- **Event ID** (foreign key): which event this harm occurred in [82].
- **ProtectedInterestType ID**: which abstract interest was harmed (link to the interest taxonomy from layer 2) [82].
- **Bearer (Affected Entity)**: who/what holds that interest and was harmed. This could be a reference to an entity record (which might include persons, organizations, or even environmental entities like rivers). For example, bearer = "Alice (person)", or bearer = "Whanganui River (as person)".
- **Effect**: a descriptor of what happened to the interest – e.g. "reduced", "removed/extinguished", "threatened", "violated". This indicates severity or type of damage. For instance, reputation might be "tarnished", a bodily interest might be "injured (physical harm)", a privacy interest might be "violated (confidential info disclosed)". This can be an enum or descriptor to aid analysis of harm impact.
- **WrongType ID (interpretation context)**: which WrongType this harm is being viewed under [83]. This field allows the same harm event to be evaluated in multiple legal frames. For instance, if a person is harmed, one HarmInstance might record it under "Assault (criminal wrong)" and another under "Tortious battery (civil wrong)". Or if a river and a person are both harmed in one event, each has its own HarmInstance row, possibly even under different WrongTypes (environmental harm vs personal injury) [83]. Essentially, it tags the harm with the legal lens through which we're interpreting it.

Using HarmInstance, we can represent **multiple victims or interest-bearers in a single event** without duplicating the event record [84]. For example, a factory explosion event could have one HarmInstance for a worker injured (interest = physical integrity of person) and another for a river polluted (interest = ecological integrity of environment). Both link back to the same event, but separately identify the harmed interest and bearers. And each can point to the appropriate WrongType (the worker's harm might tie to a negligence WrongType; the river's harm might tie to an environmental wrong or perhaps the same negligence WrongType if that covers environmental damage too).

This separation of event and harm is critical for **trauma-informed analysis**: it acknowledges that a single incident can have diverse impacts (personal trauma, community harm, environmental damage) which might need distinct redress [85]. It also avoids biasing the data model toward a single victim or a single cultural notion of harm.

TiRCorder integration: In the context of the ITIR (Intergenerational Trauma-Informed Identity Rebuilder) suite, TiRCorder is used to record events (often via voice) in a secure, user-friendly way [76]. The SensibLaw database would be the next step – where those recorded narratives are analyzed and structured. When a user records an incident in TiRCorder, it can be ingested as an Event in this schema. Natural Language Processing (NLP) can then auto-tag potential WrongTypes (filling the Event–WrongType links) and identify described harms (creating HarmInstances). Because the ontology is flexible, TiRCorder can accommodate tagging **tort events, Indigenous wrongs, human-rights abuses – all through the same schema** [86]. The result is that personal stories captured in TiRCorder aren't just raw audio/text; they become queryable data in a knowledge graph, linked to legal concepts and cultural values. This empowers both the users and analysts to see patterns, run reasoning (e.g., "has this type of harm been frequently reported?"), and ensure **no aspect of harm (physical, emotional, cultural) is overlooked** – since harm instances can be recorded for each dimension of impact.

Schema Relationships Summary

Bringing it all together, here's how the major entities connect (see also the figure above):

- **LegalSystem** – defines the jurisdiction or context. *Foreign keys:* referenced by WrongType and LegalSource (each WrongType and LegalSource belongs to one LegalSystem) [13] [15].
- **NormSourceCategory** – classifies source types. *Foreign keys:* referenced by WrongType and LegalSource (each has a primary source category) [13] [11].
- **LegalSource** – a specific law/case. Linked to WrongType via **WrongType–Source Link** records. *Foreign keys:* LegalSource has LegalSystem ID & Category ID; WrongType–Source Link stores LegalSource ID and WrongType ID [9] [21].
- **WrongType** – the central node for a wrong. *Foreign keys:* WrongType has LegalSystem ID and NormSourceCategory ID (and links out to multiple others). It connects **upwards** to its system and sources, and **downwards** to the following:
- **ProtectedInterestType** (via an association): lists which interests it protects [30].
- **ActorClass/RoleMarker** (via party constraint records): defines allowed actor and victim types [43].
- **RelationshipBasis/Shape/Intensity** (via relationship constraint records): defines required relationship between actor and victim if any [40].
- **Norm (Hohfeldian)**: built from WrongType's roles, actions, etc. Provides the duties/rights structure [51].
- **WrongElement**: elements of the cause of action (each WrongElement has WrongType ID) [63].
- **WrongElementRequirement**: linked to specific WrongElement(s), may reference ValueDimensions or other conditions [64].
- **FactorType & WrongTypeFactorPattern**: WrongTypeFactorPattern (with WrongType ID) links relevant FactorTypes to this WrongType [68].
- **RemedyModality**: via an association table, lists remedy types available for this WrongType [75].
- **ProtectedInterestType** – defines an interest (with links to ValueDimension and CulturalRegister) [29]. *Foreign keys:* ProtectedInterestType has ValueDimension ID and optional CulturalRegister ID. It is referenced by WrongTypes (many-to-many) and by HarmInstances [30] [82].
- **ValueDimension & CulturalRegister** – used by ProtectedInterestType to detail the interest's nature [24] [26]. They are reference tables (small vocabularies).
- **ActorClass, RoleMarker, RelationshipBasis, Shape, Intensity** – reference tables used to specify WrongType constraints [87] [88]. These are referenced in constraint linking tables for each WrongType (no direct link to Event; they influence which events can count as that WrongType).
- **Event** – a real-world incident. *Foreign keys:* references (or contains) context info like time, location. Links to WrongType via Event–WrongType Link [79], and to ProtectedInterestType (indirectly through HarmInstances) [82].
- **Event–WrongType Link** – join table linking Event ID and WrongType ID (many-to-many). Each Event can link multiple WrongTypes and vice versa [79].
- **HarmInstance** – ties Event to ProtectedInterestType and WrongType in a specific harm. *Foreign keys:* Event ID, ProtectedInterestType ID, WrongType ID, plus identifying the harmed entity [81]. This connects the instance layer back up to both the interest taxonomy and the wrong taxonomy.

All these relationships use **stable internal IDs**. For example, a LegalSource record has its ID referenced in the WrongType–Source link, and a WrongType's ID is used in Event–WrongType link, etc. This not only enforces referential integrity (so we can't refer to a WrongType or LegalSource that doesn't exist), but also means the model is resilient to changes in naming. A law can be renamed or re-numbered; we update the LegalSource's metadata but the links by ID still hold. A WrongType could even be deprecated

and replaced by another; historical events tagged to it still reference the correct (now archived) WrongType record, which can carry a flag if needed.

Design Pitfalls & Anti-Patterns Avoided

During development, several potential modeling mistakes were identified and consciously avoided. These "anti-patterns" and their solutions are instructive for developers extending or using the ontology:

1. **Treating specific Western tort categories as universal ontology nodes.**
   Mistake: Starting with a fixed list of torts (negligence, trespass, nuisance, etc.) as the primary schema.
   Problem: This would be overly Anglo-centric and fail to accommodate Indigenous, religious, civil law or human-rights wrongs [89]. It would force concepts to fit into Western labels, hiding the true nature of non-Western wrongs and making comparison hard [90].
   Our Solution: We designed **WrongType** as a generic frame for any "norm-violation pattern" [91]. Western torts are just particular instances of WrongType alongside mana-based harms, Islamic wrongs, etc. All share a common structure (actors, act, interest, harm, norm, remedy) with no inherent hierarchy that privileges torts [17]. This allows equal representation in the knowledge graph and easy extension when new types of wrongs are recognized.

2. **Baking "protected interests" into the WrongType definitions.**
   Mistake: Defining wrongs in terms of specific harms (e.g. having separate wrong types for "negligence causing physical injury" vs "negligence causing economic loss", or making "harm to mana" a completely distinct category) [22].
   Problem: This leads to a combinatorial explosion of wrong types and fails to capture when a single wrong implicates multiple interests (e.g. one act injures both body and reputation) [31]. It also limits comparability (you can't easily say two systems both protect "reputation" if one system buried that concept inside a specific tort).
   Our Solution: We **decomposed interests into atomic parts** – ValueDimensions and CulturalRegisters – and modeled ProtectedInterestType separately [23]. WrongTypes *link to* interests rather than contain them. This way, one WrongType can link to several interests, and two very different WrongTypes might link to a common interest (enabling analysis like "both defamation and a certain Indigenous wrong protect the value of *status.reputation*, albeit with different cultural registers") [30]. It also cleanly handles multi-interest events by using multiple HarmInstances without redefining the wrong itself.

3. **Over-encoding specific object types or sacred entities as fixed categories.**
   Mistake: Treating every culturally specific object of harm (rivers, sacred sites, ancestral lands, etc.) as distinct categories at the schema level (e.g. making "RIVER" or "TERRITORY" a fundamental type of wrong or harm) [92].
   Problem: There are innumerable culturally unique entities; you cannot enumerate them all without making the schema unwieldy or biased to certain cultures [93]. It also confuses physical object type with legal status (a river might be a physical object but also a legal person in some systems) [94].
   Our Solution: We **decoupled the concepts**:

4. We have a general **object kind** classification (e.g. "natural feature" vs "person's body" vs "artifact"),
5. A **legal personhood status** concept if needed (e.g. an "environmental entity as person" flag),

6. And a separate **Entity** table for actual instances (the specific river's identity) [95].
   In other words, the ontology can capture that something is a river and that it's considered a legal person, but these are attributes/data, not separate hardcoded subclasses. The ProtectedInterestType "ecological integrity of Country" can cover rivers, mountains, forests, etc., without needing a dozen separate interest types for each sacred object. This keeps the schema atomic and adaptable [95].

7. **Encoding complex relationships as single monolithic types.**
   Mistake: Creating one field or enum that tries to capture an entire relationship in one name (e.g. a single value "KINSHIP_GUARDIAN_VERTICAL_STRONG" to represent that whole concept) [96].
   Problem: That approach does not scale; the number of combinations of relationship basis, role, power dynamic, etc., would explode [97]. It also makes it impossible to partially match or compare relationships (you can't easily say "this scenario is kinship-based" if "kinship" is mashed together with other attributes in a code) [97]. New subtle differences would require new enum values, leading to rigidity or oversimplification [98].
   Our Solution: We **factored relationships into multiple small tables** – ActorClass, RoleMarker, RelationshipBasis, RelationshipShape, RelationshipIntensity – which can be *combined as needed* [46]. The WrongType constraints then reference whichever pieces are relevant. This yields a highly expressive model with minimal primitive categories [33]. For example, if a new scenario requires distinguishing "horizontal kinship" from "vertical kinship", we already have the dimensions to do so (same basis = kinship, different shape). We don't need a new hardcoded type, just different values in the existing fields. This modular design is much easier for developers to maintain and extend.

8. **Treating Indigenous, religious, or non-Western wrongs as special-case add-ons.**
   Mistake: Building the schema primarily for common law or Western concepts and then tacking on others via ad-hoc tables or flags (e.g. an "IndigenousWrong" table separate from "Tort", etc.) [99].
   Problem: This would embed a bias (the Western model as default, others as deviations). It makes cross-system comparison hard and would likely require duplicating logic for the "other" category. It could also lead to a silo effect where data of one type can't relate to data of another (defeating the purpose of a unified knowledge graph) [100].
   Our Solution: One unified ontology for all wrongs. We use the same WrongType structure for *all* systems [17]. The differences lie in the data: different LegalSystems, different CulturalRegisters, different values in ActorClass or remedies – but structurally everything plugs into the same slots [17]. A Maori concept of harm to *mana*, an Islamic concept of *qadhf*, and an English tort of defamation all map to: someone did X act to harm Y interest under Z norms. This not only avoids othering non-Western concepts, it actively facilitates **comparative analysis**. For instance, one could query "show me all wrong types across systems that protect reputation-related interests" and get results from common law defamation to Indigenous *mana* harms, because they all link to `status.reputation` or analogous interest types.

By steering clear of these pitfalls, the SensibLaw schema remains **flexible, inclusive, and extensible**. As developers, we can add new legal systems, new kinds of wrongs, or new interest dimensions without redesigning the whole database – we simply add new rows to these tables or new reference data, and link them appropriately. The ontology is built to grow and adapt as the project encompasses more areas of law or integrates new cultural perspectives.

Technical Implementation Notes and ITIR Integration

Although this specification is implementation-agnostic, developers should note how this conceptual model could be realized in practice and how it fits into the larger ITIR architecture:

- **Relational vs Graph Database:** The schema can be implemented in a traditional relational database (tables with foreign keys as described) or a graph database. In a relational DB (SQL), tables like `LegalSystem`, `WrongType`, `ProtectedInterestType`, etc., correspond directly to the entities above, and join tables (associative tables) implement many-to-many links (e.g. `WrongType_Source`, `WrongType_ProtectedInterest`, etc.). Ensuring proper indexing on foreign keys will be crucial for performance. In a graph database (e.g. Neo4j or AWS Neptune), these would be represented as labeled nodes and relationships: e.g. nodes of type `WrongType`, `LegalSource`, `Event` and edges like `(:WrongType)-[:DEFINED_BY]->(:LegalSource)` or `(:Event)-[:INSTANCE_OF]->(:WrongType)`. SensibLaw's design actually aligns with a **property graph approach**, as the platform stores legal entities (cases, provisions, concepts, actors) as nodes with edges like *cites*, *defines*, *applies*, etc. [101] [102]. For instance, LegalSources can have edges like *(:Case)-[:OVERRULES]->(:Case)* or *(:Statute)-[:AMENDS]->(:Statute)* beyond this core ontology. The decision of SQL vs graph might depend on query needs: graph databases excel at traversing relationships (like finding connected precedents) [103], while SQL might be more straightforward for transactional consistency and simpler lookups.

- **NLP and Parsing Pipeline:** SensibLaw integrates an NLP pipeline to ingest legal texts and output structured data [104] [56]. This pipeline (using spaCy, etc.) will map raw text (like a section of legislation) into Norm atoms, identify references to LegalSource, tag actors and conditions, etc. [105] [106]. As developers, you might be hooking the output of that pipeline into the database. For example, if the parser reads "A person must not sell spray paint unless licensed", it would produce a Norm (duty node) and perhaps a conditional exception node; the system would then store those in the Norm table linked to a WrongType "Selling restricted goods without license". The **deterministic logic tree** extraction ensures that every clause is captured and traceable [56] [57]. In the database, you might store the text offsets or IDs of source documents in the Norm or LegalSource tables for traceability (so each Norm knows exactly which document and clause it came from [57]). This is vital for auditability – clicking a node in the UI can highlight the corresponding source text [57].

- **ITIR Suite Integration:** SensibLaw is part of a broader ecosystem (ITIR). TiRCorder, as mentioned, feeds in events. Another component hinted in ITIR is an **"Identity Rebuilder"** which might use the knowledge graph to help users understand and reframe their experiences. Because the database is trauma-informed, it doesn't just catalog legal violations in abstract – it links them to personal and cultural contexts (e.g. "this event harmed your mana and is recognized in both your culture and in statutory law as a wrong"). The **interoperability** is achieved by aligning TiRCorder's data capture with SensibLaw's schema: e.g., TiRCorder might tag a transcript with preliminary labels (like "child" role detected, or "physical harm" detected via sentiment analysis of voice), which correspond to RoleMarker or ValueDimension entries, aiding in automatically populating the Event's structured fields. The end goal is a seamless flow: *Narrative -> Structured Event & Harm -> Legal reasoning*. SensibLaw's ontology is the backbone that makes sure nothing gets lost in translation and that the system can "make sense" of an event in legal terms without stripping away the event's human significance (thanks to features like CulturalRegister and multi-interest harms).

- **Broader Use Cases:** Apart from ITIR, developers might use this ontology for things like scenario simulation or comparative research. For instance, a simulation tool could allow a user to input a hypothetical event (layer 3) and then automatically traverse up to find what WrongTypes could apply (layer 2) and then retrieve the exact laws or precedents (layer 1) that would be relevant, building a *proof tree* for the scenario [60] [107]. Because every clause and factor is modeled, such a tool can explain *why* a given outcome might occur (e.g. "liability likely because factors A, B, C are present") in a transparent way, not a black-box AI guess [61] [108]. The structured approach also facilitates multilingual and cross-jurisdictional support – since the ontology normalizes concepts, the system can ingest laws in different languages and map them to the same underlying framework [109] [59]. For example, a French law and an English law both about privacy can be parsed and stored such that they both link to a ProtectedInterestType "privacy of personal information," enabling unified queries. This is part of SensibLaw's vision as an open legal reasoning platform that is not confined to one country [110].

In conclusion, the SensibLaw database schema provides a **conceptual blueprint** for handling legal knowledge in a structured, extensible manner. By separating layers, using abstracted link tables, and avoiding culturally biased design choices, it enables developers to build a system that is robust across jurisdictions and responsive to real-world complexity. Whether implemented in PostgreSQL or Neo4j, the key is that each table/relationship defined here corresponds to a real concept in legal reasoning – making the law *computable* and *navigable* without losing the nuance of different legal systems. With this foundation, developers can confidently build features like automated legal analysis, cross-law comparisons, and trauma-informed decision support, knowing the data model can handle the intricacies involved. The ontology is the common language that will let all parts of the ITIR suite (and external integrations) talk to each other about law, harm, and healing in a consistent way.

---

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] [96] [97] [98] [99] [100] SENSIBLAW - Ontology and database design.txt
file://file_00000000540c7206b136539b58c5f46d

[55] [56] [57] [58] [59] [101] [104] [105] [106] [109] [110] SensibLaw_ Open Legal Knowledge Graph & Reasoning Platform.pdf
file://file_00000000e88c7206a9ee9ef8dec8e7f9

[60] [61] [62] [102] [103] [107] [108] SensibLaw_ Open-Source AI for Legal Reasoning and Knowledge Graphs.pdf
file://file_000000009ad07206bb382c9e640cd4cb

[76] [77] SENSIBLAW - Feature timeline visualization.pdf
file://file_00000000da887206b534f5ef3359930e