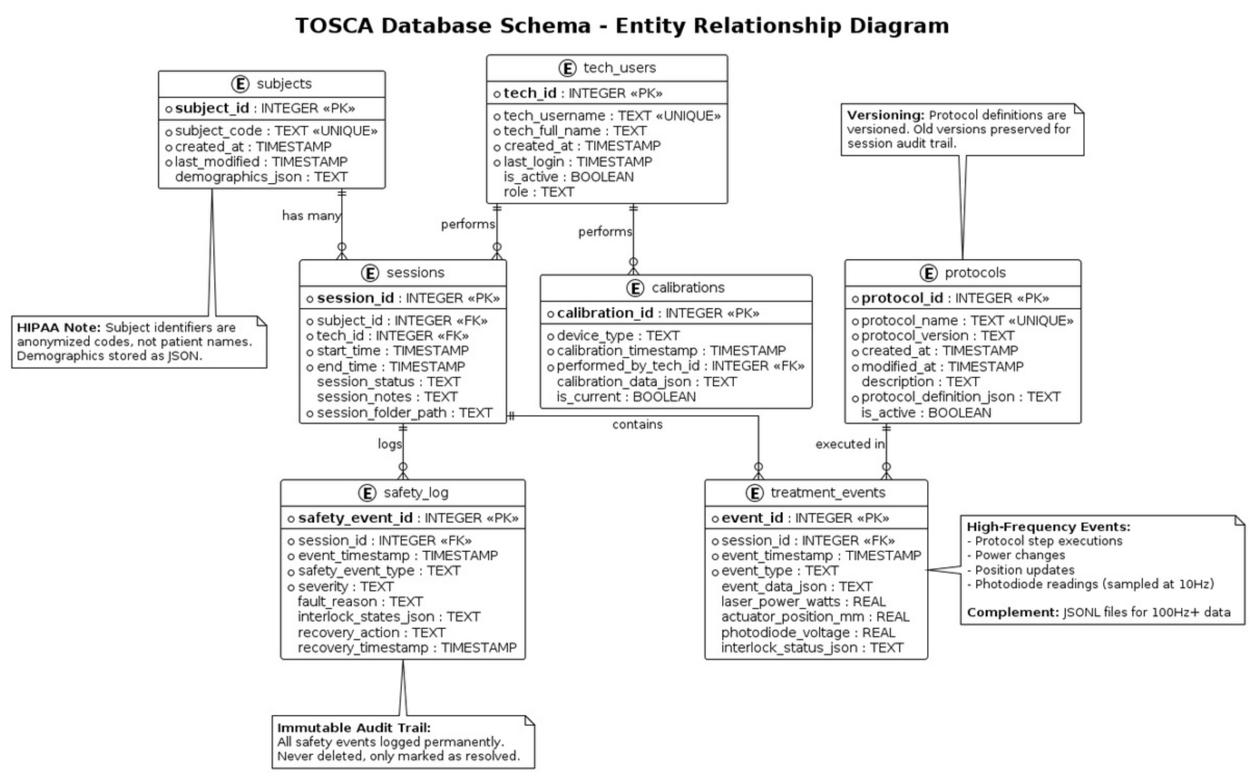
TOSCA Laser Control System - Database Schema

Architecture Diagrams

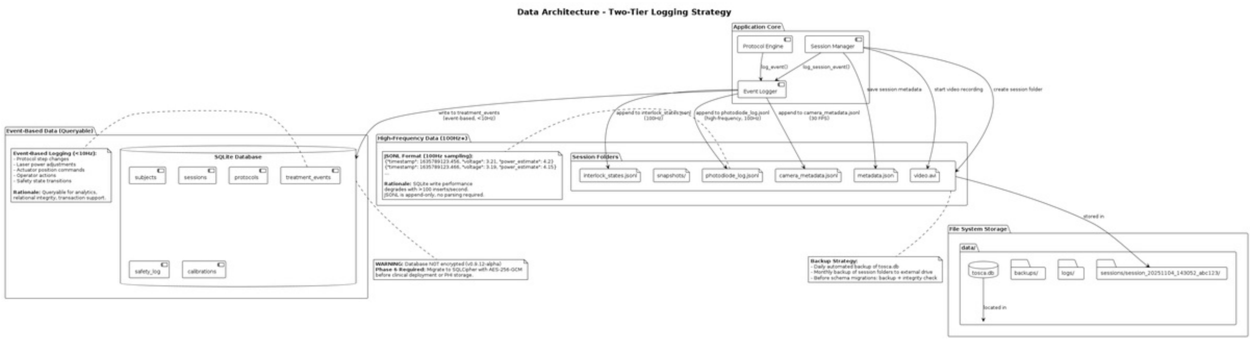
Figure 1: TOSCA Database Schema ERD



TOSCA Database Schema - Entity Relationship Diagram
subjects
subject_id : INTEGER «PK»
subject_code : TEXT «UNIQUE»
created_at : TIMESTAMP
last_modified : TIMESTAMP
demographics_json : TEXT
tech_users
tech_id : INTEGER «PK»
tech_username : TEXT «UNIQUE»
tech_full_name : TEXT
created_at : TIMESTAMP
last_login : TIMESTAMP
is_active : BOOLEAN
role : TEXT
Versioning: Protocol definitions are versioned. Old versions preserved for session audit trail.
has many
performs
performs
sessions
session_id : INTEGER «PK»
subject_id : INTEGER «FK»
tech_id : INTEGER «FK»
start_time : TIMESTAMP
end_time : TIMESTAMP
session_status : TEXT
session_notes : TEXT
session_folder_path : TEXT
calibrations
calibration_id : INTEGER «PK»
device_type : TEXT
calibration_timestamp : TIMESTAMP
performed_by_tech_id : INTEGER «FK»
calibration_data_json : TEXT
is_current : BOOLEAN
protocols
protocol_id : INTEGER «PK»
protocol_name : TEXT «UNIQUE»
protocol_version : TEXT
created_at : TIMESTAMP
modified_at : TIMESTAMP
description : TEXT
protocol_definition_json : TEXT
is_active : BOOLEAN
HIPAA Note: Subject identifiers are anonymized codes, not patient names. Demographics stored as JSON.
contains
logs
executed in
safety_log
safety_event_id : INTEGER «PK»
session_id : INTEGER «FK»
event_timestamp : TIMESTAMP
safety_event_type : TEXT
severity : TEXT
fault_reason : TEXT
interlock_states_json : TEXT
recovery_action : TEXT
recovery_timestamp : TIMESTAMP
treatment_events
event_id : INTEGER «PK»
session_id : INTEGER «FK»
event_timestamp : TIMESTAMP
event_type : TEXT
event_data_json : TEXT
laser_power_watts : REAL
actuator_position_mm : REAL
photodiode_voltage : REAL
interlock_status_json : TEXT
High-Frequency Events:
- Protocol step executions
- Power changes
- Position updates
- Photodiode readings (sampled at 10Hz)
Complement: JSONL files for 100Hz+ data
Immutable Audit Trail:
All safety events logged permanently.
Never deleted, only marked as resolved.

TOSCA Database Schema ERD

Figure 2: TOSCA Data Architecture


Data Architecture - Two-Tier Logging Strategy
Application Core
Protocol Engine
Session Manager
log_event()
log_session_event()
Event Logger
save session metadata
start video recording
create session folder
write to treatment_events (event-based, <10Hz)
append to interlock_states.jsonl (100Hz)
append to photodiode_log.jsonl (high-frequency, 100Hz)
append to camera_metadata.jsonl (30 FPS)
Event-Based Data (Queryable)
Event-Based Logging (<10Hz):
- Protocol step changes
- Laser power adjustments
- Actuator position commands
- Operator actions
- Safety state transitions
Rationale: Queryable for analytics, relational integrity, transaction support.
SQLite Database
subjects
sessions
protocols
treatment_events
safety_log
calibrations
High-Frequency Data (100Hz+)
JSONL Format (100Hz sampling):
{"timestamp": 1635789123.456, "voltage": 3.21, "power_estimate": 4.2}
{"timestamp": 1635789123.466, "voltage": 3.19, "power_estimate": 4.15}
...
Rationale: SQLite write performance degrades with >100 inserts/second. JSONL is append-only, no parsing required.
Session Folders
interlock_states.jsonl
snapshots/
photodiode_log.jsonl
camera_metadata.jsonl
metadata.json
video.avi
WARNING: Database NOT encrypted (v0.9.12-alpha)
Phase 6 Required: Migrate to SQLCipher with AES-256-GCM before clinical deployment or PHI storage.
Backup Strategy:
- Daily automated backup of tosca.db
- Monthly backup of session folders to external drive
- Before schema migrations: backup + integrity check
stored in
File System Storage
data/
tosca.db
backups/
logs/
sessions/session_20251104_143052_abc123/
located in

TOSCA Data Architecture

Document Version: 1.0 **Date:** 2025-10-15 **Database:** SQLite 3 **ORM:** SQLAlchemy (optional)

Overview

The database stores all subject information, treatment sessions, event logs, protocols, and calibration data. Design principles:

1. **Immutability**: Treatment events are write-once (no updates/deletes)
2. **Auditability**: Every change is logged with timestamp and operator
3. **Longitudinal tracking**: Subject data spans multiple sessions over time
4. **Referential integrity**: Foreign keys enforce data consistency
5. **Performance**: Indexed for common queries

Entity Relationship Diagram

```
   tech_users                subjects               protocols




                                ▼
                         ►   sessions  ◄





                   ▼                         ▼
              treatment_                session_
                events                  recordings



                                ▼
                           safety_log




  calibrations    (independent)
```

Table Definitions

1. tech_users

Stores technician/operator accounts for authentication and audit trail.

```
CREATE TABLE tech_users (
    tech_id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE NOT NULL,
    full_name TEXT NOT NULL,
    role TEXT NOT NULL,  -- 'technician', 'supervisor', 'admin'
    created_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    last_login DATETIME,
    is_active BOOLEAN DEFAULT 1,
    notes TEXT
);

-- Indexes
CREATE INDEX idx_tech_username ON tech_users(username);
CREATE INDEX idx_tech_active ON tech_users(is_active);
```

Sample Data:

```
INSERT INTO tech_users (username, full_name, role) VALUES
    ('tech01', 'Jane Doe', 'technician'),
    ('supervisor01', 'John Smith', 'supervisor');
```

2. subjects

Anonymized subject records for longitudinal tracking.

```
CREATE TABLE subjects (
    subject_id INTEGER PRIMARY KEY AUTOINCREMENT,
    subject_code TEXT UNIQUE NOT NULL,  -- Anonymized ID (e.g., 'P-2025-0001')
    date_of_birth DATE,  -- Optional, for age calculation
    gender TEXT,  -- Optional demographics
    created_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    created_by_tech_id INTEGER,
    last_modified_date DATETIME,
    notes TEXT,
    is_active BOOLEAN DEFAULT 1,
    FOREIGN KEY (created_by_tech_id) REFERENCES tech_users(tech_id)
);

-- Indexes
CREATE UNIQUE INDEX idx_subject_code ON subjects(subject_code);
CREATE INDEX idx_subject_active ON subjects(is_active);
CREATE INDEX idx_subject_created_date ON subjects(created_date);
```

Subject Code Format: `P-YYYY-NNNN` (e.g., P-2025-0001)

Sample Data:

```
INSERT INTO subjects (subject_code, created_by_tech_id, notes) VALUES
    ('P-2025-0001', 1, 'Initial consultation');
```

3. protocols

Saved treatment protocols (templates) that can be reused.

```
CREATE TABLE protocols (
    protocol_id INTEGER PRIMARY KEY AUTOINCREMENT,
    protocol_name TEXT UNIQUE NOT NULL,
    description TEXT,
    protocol_type TEXT NOT NULL,  -- 'constant', 'linear_ramp', 'multi_step', 'custom'
    protocol_data TEXT NOT NULL,  -- JSON: [{step: 1, duration: 60, power_start: 5.0, ...}, ...]
    created_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    created_by_tech_id INTEGER,
    last_modified_date DATETIME,
    last_used_date DATETIME,
    usage_count INTEGER DEFAULT 0,
    is_active BOOLEAN DEFAULT 1,

    FOREIGN KEY (created_by_tech_id) REFERENCES tech_users(tech_id)
);

-- Indexes
CREATE UNIQUE INDEX idx_protocol_name ON protocols(protocol_name);
CREATE INDEX idx_protocol_type ON protocols(protocol_type);
CREATE INDEX idx_protocol_active ON protocols(is_active);
```

Protocol Data JSON Format:

```
{
  "version": "1.0",
  "steps": [
    {
      "step_number": 1,
      "duration_seconds": 60,
      "power_start_watts": 5.0,
      "power_end_watts": 5.0,
      "ring_size_mm": 3.0,
      "ramp_type": "constant"
    },
    {
      "step_number": 2,
      "duration_seconds": 30,
      "power_start_watts": 5.0,
      "power_end_watts": 8.0,
      "ring_size_mm": 3.5,
      "ramp_type": "linear"
    }
  ],
  "safety_limits": {
    "max_power_watts": 10.0,
    "max_duration_seconds": 300
  }
}
```

4. sessions

Treatment sessions - each subject visit creates one session.

```
CREATE TABLE sessions (
    session_id INTEGER PRIMARY KEY AUTOINCREMENT,
    subject_id INTEGER NOT NULL,
    tech_id INTEGER NOT NULL,
    protocol_id INTEGER,  -- NULL if custom/ad-hoc treatment

    start_time DATETIME NOT NULL,
    end_time DATETIME,
    duration_seconds INTEGER,  -- Calculated: end_time - start_time

    status TEXT NOT NULL,  -- 'in_progress', 'completed', 'aborted', 'paused'
    abort_reason TEXT,  -- Filled if status = 'aborted'

    treatment_site TEXT,  -- Anatomical location
    treatment_notes TEXT,  -- Pre-treatment notes

    protocol_name TEXT,  -- Snapshot of protocol name (in case protocol deleted)
    protocol_data_snapshot TEXT,  -- JSON snapshot of protocol used

    session_folder_path TEXT,  -- Path to session data folder
    video_path TEXT,  -- Path to recorded video

    -- Summary statistics (calculated at end)
    total_laser_on_time_seconds REAL,
    avg_power_watts REAL,
    max_power_watts REAL,
    total_energy_joules REAL,

    post_treatment_notes TEXT,  -- Post-treatment observations
    created_date DATETIME DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (subject_id) REFERENCES subjects(subject_id),
    FOREIGN KEY (tech_id) REFERENCES tech_users(tech_id),
    FOREIGN KEY (protocol_id) REFERENCES protocols(protocol_id)
);
```

```sql
-- Indexes
CREATE INDEX idx_session_subject ON sessions(subject_id);
CREATE INDEX idx_session_tech ON sessions(tech_id);
CREATE INDEX idx_session_start_time ON sessions(start_time);
CREATE INDEX idx_session_status ON sessions(status);
CREATE INDEX idx_session_date_status ON sessions(start_time, status);
```

5. treatment_events

High-frequency event log - every significant action during treatment.

Note: For very high-frequency data (>10Hz), consider writing to JSON file first, then importing to DB post-session.

```sql
CREATE TABLE treatment_events (
    event_id INTEGER PRIMARY KEY AUTOINCREMENT,
    session_id INTEGER NOT NULL,
    timestamp DATETIME NOT NULL,
    time_offset_ms INTEGER NOT NULL,  -- Milliseconds from session start

    event_type TEXT NOT NULL,
    -- Event types:
    --   'laser_on', 'laser_off'
    --   'power_set', 'power_change'
    --   'ring_size_set', 'ring_size_change'
    --   'protocol_step_start', 'protocol_step_end'
    --   'pause', 'resume'
    --   'image_captured'
    --   'safety_warning', 'safety_trigger'
    --   'user_action'

    -- Hardware state at time of event
    laser_power_commanded_watts REAL,
    laser_power_actual_watts REAL,  -- From photodiode
    ring_size_mm REAL,
    actuator_position_um REAL,
    photodiode_voltage_v REAL,

    -- Safety status
    footpedal_state BOOLEAN,
    smoothing_device_state BOOLEAN,
    interlock_status TEXT,  -- JSON: {"footpedal": true, "smoothing": true, ...}

    -- Image/video reference
    image_path TEXT,
    video_timestamp_ms INTEGER,

    -- Additional data (JSON for flexibility)
    event_details TEXT,  -- JSON: free-form additional data

    FOREIGN KEY (session_id) REFERENCES sessions(session_id)
);

-- Indexes
CREATE INDEX idx_event_session ON treatment_events(session_id);
CREATE INDEX idx_event_timestamp ON treatment_events(timestamp);
CREATE INDEX idx_event_type ON treatment_events(event_type);
CREATE INDEX idx_event_session_time ON treatment_events(session_id, time_offset_ms);
```

Event Details JSON Examples:

```json
// laser_on event
{
  "source": "footpedal_depression",
  "protocol_step": 1
}

// safety_trigger event
{
  "trigger_source": "photodiode_out_of_range",
  "expected_power": 5.0,
  "measured_power": 2.3,
  "action_taken": "laser_shutdown"
}

// user_action event
{
  "action": "protocol_power_override",
  "old_value": 5.0,
  "new_value": 6.5,
  "reason": "subject tolerance"
}
```

6. session_recordings

Metadata for recorded video and image files.

```sql
CREATE TABLE session_recordings (
    recording_id INTEGER PRIMARY KEY AUTOINCREMENT,
    session_id INTEGER NOT NULL,
    recording_type TEXT NOT NULL,  -- 'video', 'snapshot', 'focus_stack'
    file_path TEXT NOT NULL,
    file_size_bytes INTEGER,
    duration_seconds REAL,  -- For videos
    timestamp DATETIME NOT NULL,
    frame_count INTEGER,  -- For videos
    resolution TEXT,  -- e.g., '1920x1080'
    notes TEXT,

    FOREIGN KEY (session_id) REFERENCES sessions(session_id)
);

-- Indexes
CREATE INDEX idx_recording_session ON session_recordings(session_id);
CREATE INDEX idx_recording_type ON session_recordings(recording_type);
```

7. safety_log

Comprehensive log of all safety-related events (separate from treatment_events for clarity).

```sql
CREATE TABLE safety_log (
    log_id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp DATETIME NOT NULL,
    session_id INTEGER,  -- NULL if not during active session
    tech_id INTEGER,

    event_type TEXT NOT NULL,
    -- Event types:
    --   'e_stop_pressed', 'e_stop_released'
    --   'footpedal_released', 'footpedal_timeout'
    --   'smoothing_device_fault', 'smoothing_device_recovery'
    --   'photodiode_out_of_range', 'photodiode_fault'
    --   'interlock_failure', 'interlock_recovery'
    --   'power_limit_exceeded'
    --   'system_watchdog_timeout'

    severity TEXT NOT NULL,  -- 'info', 'warning', 'critical', 'emergency'
    description TEXT NOT NULL,

    -- Hardware state at time of event
    system_state TEXT,  -- 'idle', 'ready', 'armed', 'treating', 'fault'
    laser_state TEXT,
    footpedal_state BOOLEAN,
    smoothing_device_state BOOLEAN,
    photodiode_voltage_v REAL,

    action_taken TEXT,  -- What the system did in response
    details TEXT,  -- JSON: additional data

    FOREIGN KEY (session_id) REFERENCES sessions(session_id),
    FOREIGN KEY (tech_id) REFERENCES tech_users(tech_id)
);

-- Indexes
CREATE INDEX idx_safety_timestamp ON safety_log(timestamp);
CREATE INDEX idx_safety_session ON safety_log(session_id);
CREATE INDEX idx_safety_severity ON safety_log(severity);
CREATE INDEX idx_safety_type ON safety_log(event_type);
```

8. calibrations

Device calibration data and history.

```sql
CREATE TABLE calibrations (
    calibration_id INTEGER PRIMARY KEY AUTOINCREMENT,
    calibration_date DATETIME NOT NULL,
    tech_id INTEGER NOT NULL,

    device_type TEXT NOT NULL,  -- 'laser', 'actuator', 'photodiode', 'camera'
    device_serial_number TEXT,

    calibration_type TEXT NOT NULL,
    -- Calibration types:
    --   'actuator_position_to_ring_size'  -- Maps actuator µm to ring mm
    --   'photodiode_voltage_to_power'     -- Maps voltage to laser watts
    --   'laser_power_verification'        -- Verifies commanded vs actual power
    --   'camera_focus_calibration'        -- Focus score reference
    --   'ring_detection_calibration'      -- Circle detection tuning

    calibration_data TEXT NOT NULL,  -- JSON: calibration parameters
    validation_data TEXT,  -- JSON: validation results

    is_active BOOLEAN DEFAULT 1,  -- Most recent calibration is active
    notes TEXT,

    FOREIGN KEY (tech_id) REFERENCES tech_users(tech_id)
);

-- Indexes
CREATE INDEX idx_calib_device ON calibrations(device_type);
CREATE INDEX idx_calib_date ON calibrations(calibration_date);
CREATE INDEX idx_calib_active ON calibrations(is_active);
```

Calibration Data JSON Examples:

```json
// actuator_position_to_ring_size
{
  "calibration_points": [
    {"actuator_position_um": 0, "ring_size_mm": 2.0},
    {"actuator_position_um": 1000, "ring_size_mm": 3.0},
    {"actuator_position_um": 2000, "ring_size_mm": 4.0}
  ],
  "interpolation_method": "linear",
  "fit_equation": "ring_size = 2.0 + (position / 1000)"
}

// photodiode_voltage_to_power
{
  "calibration_points": [
    {"voltage_v": 0.5, "power_watts": 1.0},
    {"voltage_v": 2.5, "power_watts": 5.0},
    {"voltage_v": 5.0, "power_watts": 10.0}
  ],
  "fit_equation": "power = voltage * 2.0",
  "r_squared": 0.998
}
```

9. system_config

Application configuration and settings (key-value store).

```sql
CREATE TABLE system_config (
    config_key TEXT PRIMARY KEY,
    config_value TEXT NOT NULL,
    config_type TEXT NOT NULL,  -- 'string', 'integer', 'float', 'boolean', 'json'
    description TEXT,
    last_modified DATETIME DEFAULT CURRENT_TIMESTAMP,
    modified_by_tech_id INTEGER,

    FOREIGN KEY (modified_by_tech_id) REFERENCES tech_users(tech_id)
);

-- Sample configuration entries
INSERT INTO system_config (config_key, config_value, config_type, description) VALUES
    ('max_laser_power_watts', '10.0', 'float', 'Hard limit for laser power'),
    ('treatment_loop_frequency_hz', '10', 'integer', 'Control loop update rate'),
    ('photodiode_warning_threshold_percent', '15', 'float', 'Deviation % before warning'),
    ('photodiode_fault_threshold_percent', '30', 'float', 'Deviation % before fault'),
    ('video_resolution', '1920x1080', 'string', 'Recording resolution'),
    ('video_fps', '30', 'integer', 'Recording frame rate'),
    ('session_data_path', 'data/sessions', 'string', 'Base path for session data');
```

Views (Convenience Queries)

View: Active Sessions

```sql
CREATE VIEW active_sessions AS
SELECT
    s.session_id,
    s.subject_id,
    p.subject_code,
    s.tech_id,
    t.full_name as tech_name,
    s.start_time,
    s.status,
    s.protocol_name,
    strftime('%s', 'now') - strftime('%s', s.start_time) as duration_seconds
FROM sessions s
JOIN subjects p ON s.subject_id = p.subject_id
JOIN tech_users t ON s.tech_id = t.tech_id
WHERE s.status IN ('in_progress', 'paused');
```

View: Subject Session History

```sql
CREATE VIEW subject_session_history AS
SELECT
    p.subject_id,
    p.subject_code,
    s.session_id,
    s.start_time,
    s.end_time,
    s.status,
    s.protocol_name,
    s.total_laser_on_time_seconds,
    s.avg_power_watts,
    s.total_energy_joules,
    t.full_name as technician
FROM subjects p
LEFT JOIN sessions s ON p.subject_id = s.subject_id
LEFT JOIN tech_users t ON s.tech_id = t.tech_id
ORDER BY p.subject_id, s.start_time DESC;
```

View: Safety Events Summary

```sql
CREATE VIEW safety_events_summary AS
SELECT
    DATE(timestamp) as event_date,
    severity,
    event_type,
    COUNT(*) as event_count
FROM safety_log
GROUP BY DATE(timestamp), severity, event_type
ORDER BY event_date DESC, severity;
```

Common Queries

1. Find Subject by Code

```sql
SELECT * FROM subjects
WHERE subject_code = 'P-2025-0001';
```

2. Get All Sessions for Subject

```sql
SELECT
    s.session_id,
    s.start_time,
    s.end_time,
    s.status,
    s.protocol_name,
    s.total_energy_joules,
    t.full_name as technician
FROM sessions s
JOIN tech_users t ON s.tech_id = t.tech_id
WHERE s.subject_id = ?
ORDER BY s.start_time DESC;
```

3. Get Treatment Events for Session

```sql
SELECT
    timestamp,
    event_type,
    laser_power_commanded_watts,
    laser_power_actual_watts,
    ring_size_mm,
    photodiode_voltage_v
FROM treatment_events
WHERE session_id = ?
ORDER BY time_offset_ms;
```

4. Get Active Calibration for Device

```sql
SELECT calibration_data
FROM calibrations
WHERE device_type = 'photodiode'
  AND is_active = 1
ORDER BY calibration_date DESC
LIMIT 1;
```

5. Safety Events in Last 24 Hours

```sql
SELECT
    timestamp,
    event_type,
    severity,
    description,
    session_id
FROM safety_log
WHERE timestamp >= datetime('now', '-1 day')
ORDER BY timestamp DESC;
```

6. Protocol Usage Statistics

```sql
SELECT
    protocol_name,
    usage_count,
    last_used_date,
    (SELECT COUNT(*) FROM sessions WHERE protocol_id = p.protocol_id) as actual_usage
FROM protocols p
WHERE is_active = 1
ORDER BY usage_count DESC;
```

Database Initialization Script

```sql
-- Enable foreign key constraints
PRAGMA foreign_keys = ON;

-- Set journal mode for better concurrency
PRAGMA journal_mode = WAL;

-- Create all tables (in dependency order)
-- ... (include all CREATE TABLE statements above) ...

-- Create views
-- ... (include all CREATE VIEW statements above) ...

-- Insert default tech user
INSERT INTO tech_users (username, full_name, role) VALUES
    ('admin', 'System Administrator', 'admin');

-- Insert default system config
-- ... (include system_config INSERTs) ...

-- Log database creation
INSERT INTO safety_log (timestamp, event_type, severity, description, system_state, action_taken)
VALUES (CURRENT_TIMESTAMP, 'database_initialized', 'info', 'Database schema created', 'idle', 'none');
```

Migration Strategy

Use **Alembic** for schema migrations:

```python
# alembic/versions/001_initial_schema.py
def upgrade():
    # Create tables
    pass

def downgrade():
    # Rollback
    pass
```

Versioning convention: `YYYYMMDD_NNN_description.py`

Backup Strategy

1. **Automatic backups**: Before each session, backup database
2. **Backup location**: `data/backups/laser_control_YYYYMMDD_HHMMSS.db`
3. **Retention**: Keep 30 days of daily backups
4. **Session data**: Backed up separately (video files are large)

Performance Considerations

1. **Indexing**: All foreign keys and frequently queried columns indexed
2. **Partitioning**: Consider separate tables for archived sessions (>1 year old)
3. **Write optimization**: Use transactions for bulk inserts (treatment_events)
4. **Query optimization**: Use EXPLAIN QUERY PLAN to optimize slow queries

Data Integrity

1. **Foreign key constraints**: Enabled (PRAGMA foreign_keys = ON)
2. **Write-ahead logging**: Enabled (PRAGMA journal_mode = WAL)
3. **Validation**: Enforce in application layer before DB write
4. **Immutability**: treatment_events and safety_log are append-only

---

Document Owner: Database Architect **Last Updated:** 2025-10-15