Open Chess Software Architecture

William Stenberg

2020-04-23
Version 0.2

Introduction

This document outlines the software architecture for the web-based application *Open Chess* that offers an interactive chess board through a React Application served with Flask. Using the application (playing a game of chess through mouse interaction), the user is encouraged to explore the early-game strategies of the game of chess. The backend of the application (written in Flask, with a MongoDB database) internally evaluates positions and assigns them scores. The evaluation process is performed by Stockfish, a chess computer engine. Stockfish is used to assign scores to board positions fed to it, representing a move's soundness in the game. The scores for the initial and resulting board states in a move form the weight of the move, taking into account other avaliable moves in the initial position.

The application defines *theory*, which are moves in certain positions of the board that are known to be good, typically generated by parsing a precompiled set of games in the Polyglot format, or by reading in PGN files of Grandmaster games. Moves that are known by the application are shown as arrows color-coded by their score: A good move is green, a bad move is red, and a theory move is blue. Their score as evaluated by Stockfish are available as arrow tooltips. Moves that have been added (through parsing Polyglot or PGN) but not yet evaluated are white.

Making a move, by dragging and dropping the pieces in the frontend client, updates the position with a fetch call to the backend. Making an unknown move will trigger an analysis of the position, showing a loading spinner during the few seconds of analysis time.

The app is intended for use as a training tool, prompting the player to make good moves in pre-determined board positions. In the future, more features are planned. Most notably, the author intends to add a game-mode in which the user is prompted without arrow help to make the best move in any given position, thus training their opening repertoire.

The Open Chess source code is on `https://github.com/WilliamStenberg/open-chess`, and is Free Software.

Database layout

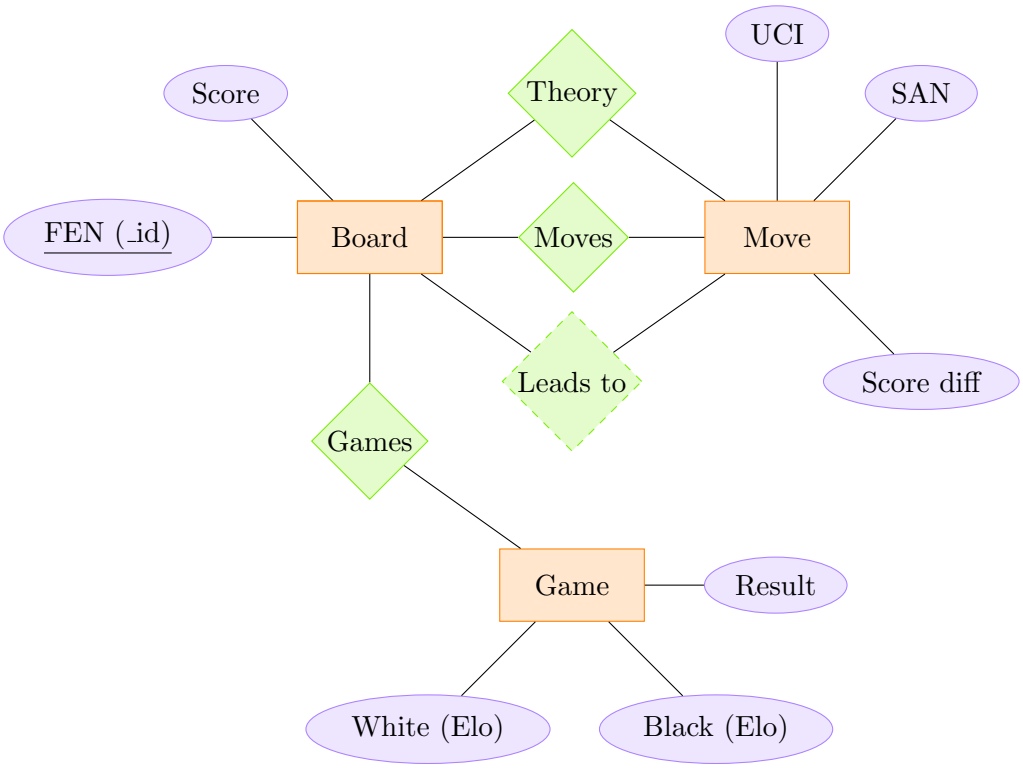

Figure 1: Entity-Relationship (ER) diagram for *open-chess* database

The entity-relationship diagram in Figure 1 outlines the structure of the database, implemented in SQLAlchemy in the backend of *open-chess*. The method of drawing the diagram comes from `https://www.guitex.org/home/images/ArsTeXnica/AT015/drawing-ER-diagrams-with-TikZ.pdf`. Below, each of the elements are explained in terms of their intended use in the application:

- **Board**: Representing a chess board position, i.e. a configuration of pieces. The board is populated with known moves, and when analysed also a score. When adding games by PGN, boards can also hold information about what games they were seen in. This is interesting to those trying to study certain strong players.
  - **FEN** (index): A board is uniquely defined by its FEN string (Forsyth-Edwards Notation) which is computed by taking into account piece positions and encodes which player is to move next. This is used as the index, allowing for quick lookups.

  - **Score**: A board's score is its numeric evaluation by a chess engine - who's winning. A score is computed from a player's perspective to a centipawn (hundredths of a pawn-up-advantage). A positive score with White to move indicates their advantage, as does a positive score for Black with Black to move.
  - **Moves**: An array of objects representing legal moves from the board position. Contains the move's UCI, SAN, score difference, and the board the move leads to, by FEN.
  - **Theory**: An array of Move objects that are separated to be identified as theory moves, verified by high-level play.
  - **Games**: A list of ObjectIDs to Game objects, see below.

- **Move**: Not in its own collection in the database, these objects are used to transition between boards.
  - **UCI**: The UCI representation of the move, comprised of the square the moved piece starts and lands on, concatenated to a four-letter string, e.g. "e2e4", "b1c3".
  - **SAN**: The SAN representation of the move, which is more commonly used by humans when talking about moves, e.g. "e4", "Nc3".
  - **Score difference**: Centipawn difference between board positions before and after the move. A positive score difference indicates a sound move, while a negative means the move might be a mistake or even a blunder.
  - **Leads to**: FEN of the resulting board after the move is made; The relationship between a move and the board it leads to. This is used to traverse the tree of moves and boards, using lookings on the Board collection by FEN.

- **Game**: A subset of PGN game information. Contains the black and white players and their respective Elo ratings, and the outcome of the game as a string out of "1 - 0", "0 - 1", and "1/2 - 1/2".