Database System Architecture

INFO-H-417

Group:
Sony Shrestha
Aayush Paudel
Hadi El Yakhni Iyoha
Peace Osamuyi

Professor:
Mahmoud SAKR
Maxime Schoemans

Agenda

01 Overview

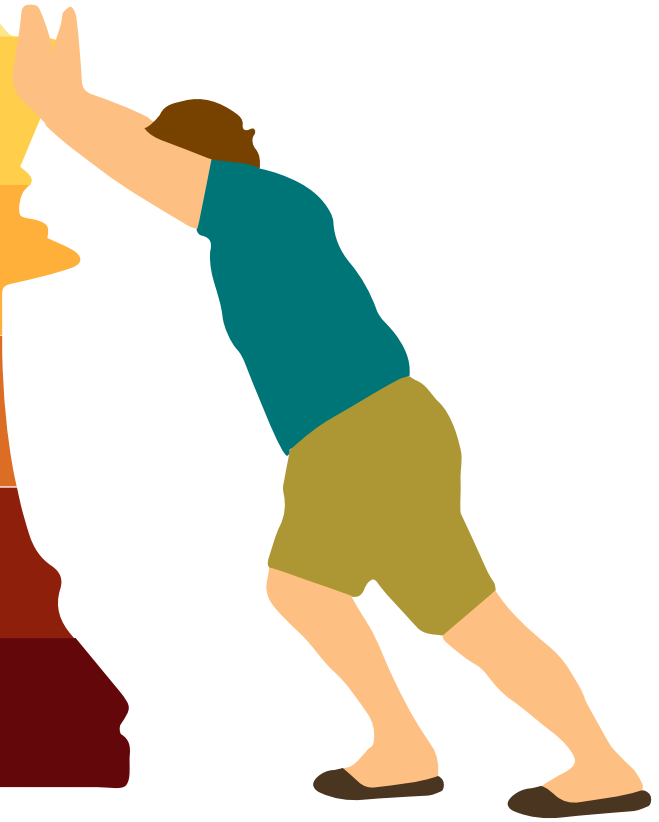
02 Data Type

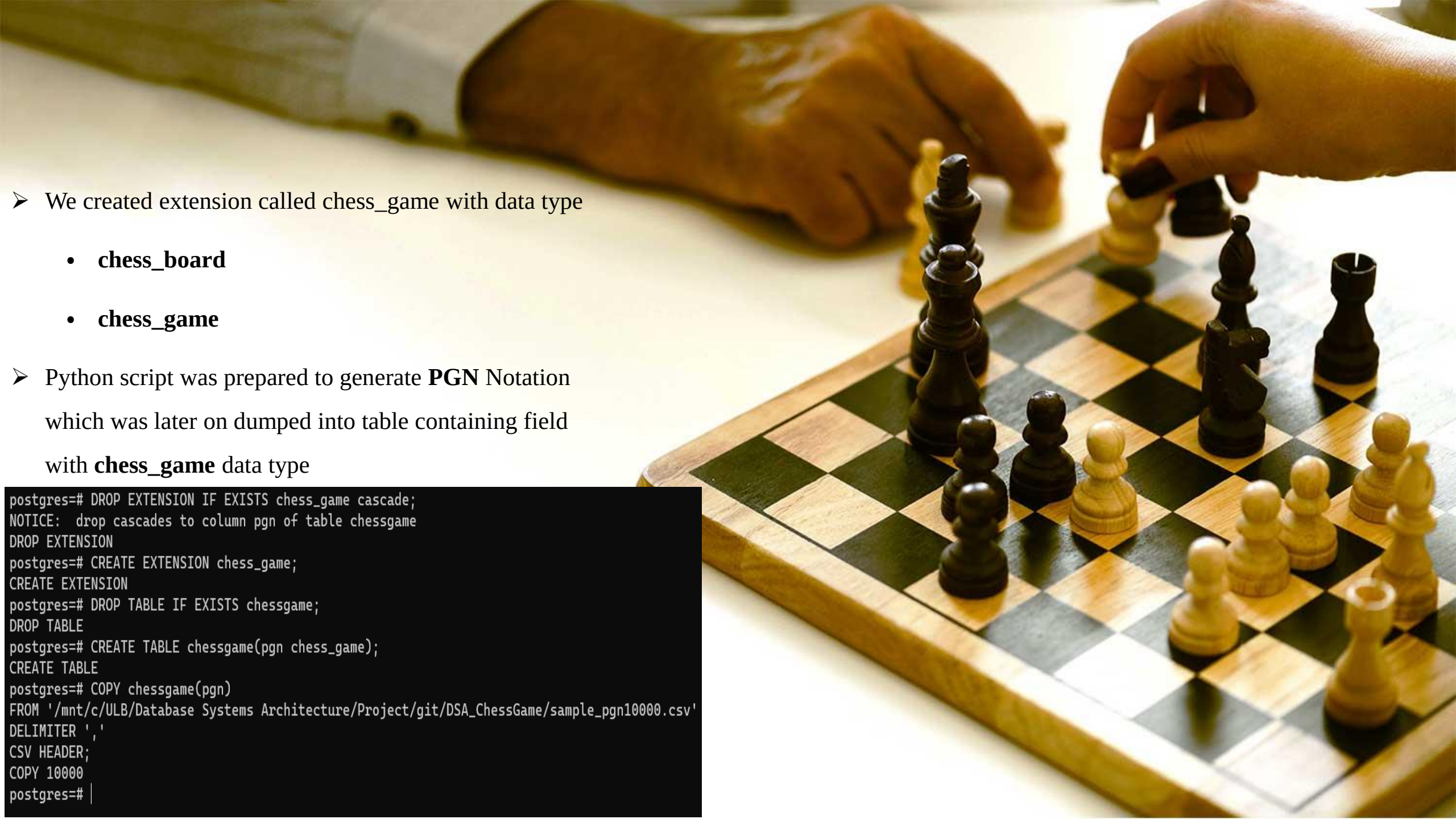
03 Function

04 Index

05 Validation

06 Conclusion





- We created extension called chess_game with data type
  - **chess_board**
  - **chess_game**
- Python script was prepared to generate **PGN** Notation which was later on dumped into table containing field with **chess_game** data type

```
postgres=# DROP EXTENSION IF EXISTS chess_game cascade;
NOTICE:  drop cascades to column pgn of table chessgame
DROP EXTENSION
postgres=# CREATE EXTENSION chess_game;
CREATE EXTENSION
postgres=# DROP TABLE IF EXISTS chessgame;
DROP TABLE
postgres=# CREATE TABLE chessgame(pgn chess_game);
CREATE TABLE
postgres=# COPY chessgame(pgn)
FROM '/mnt/c/ULB/Database Systems Architecture/Project/git/DSA_ChessGame/sample_pgn10000.csv'
DELIMITER ','
CSV HEADER;
COPY 10000
postgres=#
```

Data Type: Chess Board

- **SCL_boardFromFEN** converts **FEN** to **chess_board** data type
- **SCL_boardToFEN** converts **chess_board** data type to **FEN**

```
// Input Function for chess_board Data Type
PG_FUNCTION_INFO_V1(chess_board_in);
Datum chess_board_in(PG_FUNCTION_ARGS)
{
    char *str = PG_GETARG_CSTRING(0);

    SCL_Board *board = palloc(sizeof(SCL_Board));

    if (SCL_boardFromFEN(board, str) == 0)
        ereport(ERROR, (errcode(ERRCODE_INVALID_PARAMETER_VALUE), errmsg("Invalid board state")));

    PG_FREE_IF_COPY(str, 0);
    PG_RETURN_BOARD_P(board);
}


// Output Function for chess_board Data Type
PG_FUNCTION_INFO_V1(chess_board_out);
Datum chess_board_out(PG_FUNCTION_ARGS)
{
    SCL_Board *board = PG_GETARG_BOARD_P(0);

    char str[256];

    if (SCL_boardToFEN(board, str) == 0)
        ereport(ERROR, (errcode(ERRCODE_INVALID_PARAMETER_VALUE), errmsg("invalid board state")));

    PG_RETURN_CSTRING(str);
}
```

```
CREATE OR REPLACE FUNCTION chess_board_in(cstring)
  RETURNS chess_board
  AS 'MODULE_PATHNAME'
  LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;


CREATE OR REPLACE FUNCTION chess_board_out(chess_board)
  RETURNS cstring
  AS 'MODULE_PATHNAME'
  LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;


CREATE TYPE chess_board (
  internallength = 69, -- 69 is the length of the `board` type from `smallchesslib.h`
  input          = chess_board_in,
  output         = chess_board_out
);
```

Data Type: Chess Game

- **SCL_recordFromPGN** converts **PGN** to **chess_game** data type
- **SCL_printPGN** converts **chess_game** data type to **PGN**

```
CREATE OR REPLACE FUNCTION chess_game_in(cstring)
  RETURNS chess_game
  AS 'MODULE_PATHNAME'
  LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;


CREATE OR REPLACE FUNCTION chess_game_out(chess_game)
  RETURNS cstring
  AS 'MODULE_PATHNAME'
  LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;


CREATE TYPE chess_game (
  internallength = 512, -- 512 is the length of the `record` type from `smallchesslib.h`
  input          = chess_game_in,
  output         = chess_game_out
);
```

```
// Input Function for chess_game Data Type
PG_FUNCTION_INFO_V1(chess_game_in);
Datum chess_game_in(PG_FUNCTION_ARGS)
{
    char *str = PG_GETARG_CSTRING(0);

    SCL_Record *record = palloc(sizeof(SCL_Record));

    SCL_recordFromPGN(record, str);

    PG_FREE_IF_COPY(str, 0);
    PG_RETURN_GAME_P(record);
}


// Output Function for chess_game Data Type
PG_FUNCTION_INFO_V1(chess_game_out);
Datum chess_game_out(PG_FUNCTION_ARGS)
{
    char *game = palloc(sizeof(char) * 4096);

    SCL_Record *record = PG_GETARG_GAME_P(0);
    SCL_printPGN(record, putCharStr, 0);
    for (int i = 0; i < 4096; i++)
    {
        if (str[i] == '*')
            game[i] = '\0';
        else
            game[i] = str[i];
    }
    str[0] = '\0';
    PG_RETURN_CSTRING(game);
}
```

Function: getBoard

- **getBoard(chessgame, integer) -> chessboard**
- Returns the board state at a given half-move
- **SCL_recordApply** is used to get board state after applying n half moves in given chess game

```c
// Function to return chess_board after apply n half_move in given chess_game
SCL_Board *get_board_internal(SCL_Record record, int half_moves)
{
    SCL_Board *board = get_starting_board();
    SCL_recordApply(record, board, half_moves);
    char *str=chess_board_to_str(board);
    pfree(str);
    return board;
}


// Function to get chess_board after apply n half_move in given chess_game
PG_FUNCTION_INFO_V1(getBoard);
Datum getBoard(PG_FUNCTION_ARGS)
{
    SCL_Record *record = PG_GETARG_GAME_P(0);
    int half_move = PG_GETARG_INT32(1);

    SCL_Board *board = get_board_internal(*record, half_move);

    PG_RETURN_BOARD_P(board);
}
```

```sql
CREATE FUNCTION getBoard(chess_game, integer)
    RETURNS chess_board
    AS 'MODULE_PATHNAME', 'getBoard'
    LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;
```

Function: getFirstMoves

- **getFirstMoves(chessgame, integer) -> chessgame**
- Returns the chessgame truncated to its first N half-moves
- Helper function **truncate_pgn_internal** extracts subset of chess game, given the number of half moves

```
CREATE FUNCTION getFirstMoves(chess_game, integer)
  RETURNS chess_game
  AS 'MODULE_PATHNAME', 'getFirstMoves'
  LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;
```

```
// Function to get first n half moves from chess_game
PG_FUNCTION_INFO_V1(getFirstMoves);
Datum getFirstMoves(PG_FUNCTION_ARGS)
{
    SCL_Record *record = PG_GETARG_GAME_P(0);
    int half_move = PG_GETARG_INT32(1);
    char *chess_game_str;
    char result_pgn[256];
    SCL_Record *output_record = palloc(sizeof(SCL_Record));

    chess_game_str = chess_game_to_str(record);
    truncate_pgn_internal(chess_game_str, half_move, result_pgn);
    SCL_recordFromPGN(output_record, result_pgn);
    PG_RETURN_GAME_P(output_record);
}
```

```
void truncate_pgn_internal(char *chess_notation, int half_moves, char *result_board)
{
    int curr_str_idx = 0;
    int curr_half_move = 0;

    if (half_moves == 0)
    {
        result_board[0] = '\0';
        return;
    }

    while (1)
    {
        // copy the full move number and the space after it
        while (1)
        {
            result_board[curr_str_idx++] = chess_notation[curr_str_idx];
            if (result_board[curr_str_idx - 1] == ' ')
                break;
        }

        // copy the first move
        while (1)
        {
            result_board[curr_str_idx++] = chess_notation[curr_str_idx];
            if (result_board[curr_str_idx - 1] == ' ')
                break;
        }

        if (++curr_half_move == half_moves)
            break;

        // copy the second move
        while (1)
        {
            result_board[curr_str_idx++] = chess_notation[curr_str_idx];
            if (result_board[curr_str_idx - 1] == ' ' || result_board[curr_str_idx - 1] == '\0')
                break;
        }

        if (++curr_half_move == half_moves || result_board[curr_str_idx - 1] == '\0')
            break;
    }

    result_board[curr_str_idx] = '\0';
}
```

Function: hasOpening

- **hasOpening(chessgame, chessgame) -> boolean**
- Returns true if the first chess game starts with the exact same set of moves as the second chess game
- Logic for this function has been implemented in SQL

```
CREATE OR REPLACE FUNCTION hasOpening(game chess_game, opening chess_game)
    RETURNS boolean
    AS $$
        select game >= opening AND game < opening;
    $$
    LANGUAGE sql;
```

Function: hasBoard

- **hasBoard(chessgame, chessboard, integer) -> boolean**

- Returns true if the chessgame contains the given board state in its first N half-moves without considering the move count, castling right, en passant pieces.

- Logic for this function has been implemented in SQL

```
CREATE OR REPLACE FUNCTION hasBoard(game chess_game, board chess_board,half_move integer)
    RETURNS boolean
    AS $$
        select game @@ CONCAT(half_move::VARCHAR,',',board::VARCHAR);
    $$
    LANGUAGE sql;
```

Index: B-Tree

For implementation of B-Tree, we created

- five operators <, <=, =, >, >=
- five functions to implement above operators
- Compare function to compare two chess games

```
CREATE OPERATOR CLASS chess_game_ops
DEFAULT FOR TYPE chess_game USING btree
AS
    OPERATOR        1       < ,
    OPERATOR        2       <= ,
    OPERATOR        3       = ,
    OPERATOR        4       >= ,
    OPERATOR        5       > ,
    FUNCTION        1       chess_game_cmp(chess_game, chess_game);
```

```
CREATE OPERATOR = (
  LEFTARG  = chess_game,
  RIGHTARG = chess_game,
  PROCEDURE = chess_game_eq,
  COMMUTATOR = =
);

CREATE OPERATOR < (
  LEFTARG  = chess_game,
  RIGHTARG = chess_game,
  PROCEDURE = chess_game_lt
);

CREATE OPERATOR <= (
  LEFTARG  = chess_game,
  RIGHTARG = chess_game,
  PROCEDURE = chess_game_le,
  COMMUTATOR = >=
);

CREATE OPERATOR > (
  LEFTARG  = chess_game,
  RIGHTARG = chess_game,
  PROCEDURE = chess_game_gt
);

CREATE OPERATOR >= (
  LEFTARG  = chess_game,
  RIGHTARG = chess_game,
  PROCEDURE = chess_game_ge,
  COMMUTATOR = <=
);

CREATE OR REPLACE FUNCTION chess_game_cmp(chess_game, chess_game)
  RETURNS integer
  AS 'MODULE_PATHNAME', 'chess_game_cmp'
  LANGUAGE C IMMUTABLE STRICT PARALLEL SAFE;
```

Index: B-Tree

```c
// Function to check if first chess_game is less than second one
PG_FUNCTION_INFO_V1(chess_game_lt);
Datum chess_game_lt(PG_FUNCTION_ARGS)
{
    SCL_Record *c = PG_GETARG_GAME_P(0);
    SCL_Record *d = PG_GETARG_GAME_P(1);

    char *chess_game_str1 = chess_game_to_str(c);

    char *chess_game_str2 = chess_game_to_str(d);
    strcat(chess_game_str2, "~~");

    bool result = strcmp(chess_game_str1, chess_game_str2) < 0;

    PG_FREE_IF_COPY(c, 0);
    PG_FREE_IF_COPY(d, 1);
    PG_RETURN_BOOL(result);
}


// Function to check if first chess_game is less than equal to second one
PG_FUNCTION_INFO_V1(chess_game_le);
Datum chess_game_le(PG_FUNCTION_ARGS)
{
    SCL_Record *c = PG_GETARG_GAME_P(0);
    SCL_Record *d = PG_GETARG_GAME_P(1);

    bool result = compare_games(c, d) <= 0;

    PG_FREE_IF_COPY(c, 0);
    PG_FREE_IF_COPY(d, 1);
    PG_RETURN_BOOL(result);
}
```

```c
// Function to check if first chess_game is equal to second one
PG_FUNCTION_INFO_V1(chess_game_eq);
Datum chess_game_eq(PG_FUNCTION_ARGS)
{
    SCL_Record *c = PG_GETARG_GAME_P(0);
    SCL_Record *d = PG_GETARG_GAME_P(1);

    bool result = compare_games(c, d) == 0;

    PG_FREE_IF_COPY(c, 0);
    PG_FREE_IF_COPY(d, 1);
    PG_RETURN_BOOL(result);
}


// Function to check if first chess_game is greater than second one
PG_FUNCTION_INFO_V1(chess_game_gt);
Datum chess_game_gt(PG_FUNCTION_ARGS)
{
    SCL_Record *c = PG_GETARG_GAME_P(0);
    SCL_Record *d = PG_GETARG_GAME_P(1);

    bool result = compare_games(c, d) > 0;

    PG_FREE_IF_COPY(c, 0);
    PG_FREE_IF_COPY(d, 1);
    PG_RETURN_BOOL(result);
}
```

```c
// Function to check if first chess_game is greater than equal to second one
PG_FUNCTION_INFO_V1(chess_game_ge);
Datum chess_game_ge(PG_FUNCTION_ARGS)
{
    SCL_Record *c = PG_GETARG_GAME_P(0);
    SCL_Record *d = PG_GETARG_GAME_P(1);

    bool result = compare_games(c, d) >= 0;

    PG_FREE_IF_COPY(c, 0);
    PG_FREE_IF_COPY(d, 1);
    PG_RETURN_BOOL(result);
}


// Function to compare two chess_game
// 0 - two chess game are equal
// -1 - first chess_game is smaller than the second one
// 1 - first chess_game is larger than the second one
PG_FUNCTION_INFO_V1(chess_game_cmp);
Datum chess_game_cmp(PG_FUNCTION_ARGS)
{
    SCL_Record *c = PG_GETARG_GAME_P(0);
    SCL_Record *d = PG_GETARG_GAME_P(1);

    int diff = compare_games(c, d);

    int result = 0;
    if (diff < 0)
        result = -1;
    else if (diff > 0)
        result = 1;

    PG_FREE_IF_COPY(c, 0);
    PG_FREE_IF_COPY(d, 1);
    PG_RETURN_INT32(result);
}
```

Index: GIN

For implementation of GIN Index, we created

- Operator @> which checks if given board state is present in chess game within n half moves
- A function to implement @> operator
- Compare function to compare two chess boards
- Other functions like chess_game_extractValue, chess_game_extractQuery, chess_game_consistent

```
// Function to compare two chess_board
PG_FUNCTION_INFO_V1(chess_board_compare);
Datum chess_board_compare(PG_FUNCTION_ARGS)
{
    SCL_Board *c = PG_GETARG_BOARD_P(0);
    SCL_Board *d = PG_GETARG_BOARD_P(1);

    char *board1=chess_board_to_str(c);
    char *board2=chess_board_to_str(d);

    int index1 = strcspn(board2, " ");
    int diff = strncmp(board1, board2, index1);

    int result = 0;
    if (diff < 0)
        result = -1;
    else if (diff > 0)
        result = 1;

    PG_FREE_IF_COPY(c, 0);
    PG_FREE_IF_COPY(d, 1);
    PG_RETURN_INT32(result);
}
```

```
CREATE OPERATOR @> (
  PROCEDURE = chess_contains_within_func,
  LEFTARG = chess_game,
  RIGHTARG = text
);


CREATE OPERATOR CLASS chessgame_ops
DEFAULT FOR TYPE chess_game USING gin AS
    OPERATOR        1       @>(chess_game,text),
    FUNCTION        1       chess_board_compare(chess_board, chess_board),
    FUNCTION        2       chess_game_extractValue(chess_game,internal),
    FUNCTION        3       chess_game_extractQuery(text,internal,internal,internal,internal,internal,internal),
    FUNCTION        4       chess_game_consistent(internal,internal,internal,internal,internal,internal),
    STORAGE chess_board;
```

Index: GIN

```c
// extractValue function for GIN Index
PG_FUNCTION_INFO_V1(chess_game_extractValue);
Datum chess_game_extractValue(PG_FUNCTION_ARGS)
{
    SCL_Record *record = PG_GETARG_GAME_P(0);
    int32 *nkeys = (int32 *) PG_GETARG_POINTER(1);
    int total_half_moves = count_half_moves(record)+1;

    // Allocate memory for an array of Datums
    Datum *boards = (Datum *) palloc(total_half_moves * sizeof(Datum));

    for (int half_move = 0; half_move < total_half_moves; half_move++) {
        SCL_Board *current_board = get_board_internal(*record, half_move);
        if (current_board) {
            boards[half_move] = PointerGetDatum(current_board);
        } else {
            boards[half_move] = (Datum)0;
        }
    }

    *nkeys = total_half_moves;
    PG_FREE_IF_COPY(record, 0);
    PG_RETURN_POINTER(boards);
}
```

```c
// extractQuery function used by GIN Index
PG_FUNCTION_INFO_V1(chess_game_extractQuery);
Datum chess_game_extractQuery(PG_FUNCTION_ARGS)
{
    text *text_str = PG_GETARG_TEXT_PP(0);
    int32 *nkeys = (int32 *) PG_GETARG_POINTER(1);
    StrategyNumber strategy = PG_GETARG_UINT16(2);
    bool **pmatch = (bool **) PG_GETARG_POINTER(3);
    Pointer *extra_data = (Pointer *) PG_GETARG_POINTER(4);
    bool **nullFlags = (bool **) PG_GETARG_POINTER(5);
    int32 *searchMode = (int32 *) PG_GETARG_POINTER(6);

    // Create an array of datum type
    Datum *boards = (Datum *) palloc(sizeof(Datum));

    // Convert second parameter text to string
    char *str = text_to_cstring(text_str);

    // String contains value in the form of half_move,chess_board
    // Tokenize string to extract value of half_move and chess_board
    char *token = strtok(str, ",");
    int check_till = atoi(token); // First token is an integer
    token = strtok(NULL, ",");
    char *board_state1=palloc(sizeof(char) * 256);  // Second toekn is a string
    strcpy(board_state1, token);

    // Convert string fen to chess_board data type
    SCL_Board *board = palloc(sizeof(SCL_Board));
    SCL_boardFromFEN(board, board_state1);
    boards[0] = PointerGetDatum(board);

    *searchMode = GIN_SEARCH_MODE_DEFAULT;
    *nkeys = 1;
    PG_FREE_IF_COPY(text_str, 0);
    PG_RETURN_POINTER(boards);
}
```

Index: GIN

```c
// consistent function for GIN Index
PG_FUNCTION_INFO_V1(chess_game_consistent);
Datum chess_game_consistent(PG_FUNCTION_ARGS)
{
    // Retrieve the array indicating which query keys are present in the indexed value
    bool *check = (bool *) PG_GETARG_POINTER(0);
    // Retrieve the strategy number
    uint16 strategy = PG_GETARG_UINT16(1);
    // Query information is not used in this example but can be used for more complex strategies
    Datum query = PG_GETARG_DATUM(2);
    // Number of keys in the query
    int32_t nkeys = PG_GETARG_INT32(3);
    // Extra data passed from the extractQuery function, not used in this example
    Pointer *extra_data = (Pointer *) PG_GETARG_POINTER(4);
    // Pointer to flag indicating whether a recheck is required
    bool *recheck = (bool *) PG_GETARG_POINTER(5);

    strategy=1;

    // Example: Simple exact matching strategy
    if (strategy == 1) {
        // Iterate over each key
        for (int i = 0; i < nkeys; i++) {
            // If any of the keys are present, the indexed value is consistent with the query
            if (check[i]) {
                *recheck = false;  // No need to recheck as the index test is deemed exact
                PG_RETURN_BOOL(true);
            }
        }
    }

    // No keys matched, or for other unhandled strategies, return false
    *recheck = false;
    PG_FREE_IF_COPY(check, 0);
    PG_FREE_IF_COPY(extra_data,4);
    PG_FREE_IF_COPY(recheck,5);
    PG_RETURN_BOOL(false);
}
```

Data Type Validation: Chess Board

- Validate creating table with chess_board data type followed by insert and select statement

- Validate type cast operation from string to chess_board data type

```
postgres=# DROP TABLE IF EXISTS chessboard;
NOTICE:  table "chessboard" does not exist, skipping
DROP TABLE
postgres=# CREATE TABLE chessboard(fen chess_board);
CREATE TABLE
postgres=# INSERT INTO chessboard
VALUES
('r2q1rk1/1b1pbppp/p4n2/1pp3B1/3pP3/PB1P4/1PP1NPPP/R2Q1RK1 b - - 1 12');
INSERT 0 1
postgres=# SELECT * FROM chessboard;
                                      fen
------------------------------------------------------------------------
 r2q1rk1/1b1pbppp/p4n2/1pp3B1/3pP3/PB1P4/1PP1NPPP/R2Q1RK1 b KQkq - 1 12
(1 row)
```

```
postgres=# SELECT 'r2q1rk1/1b1pbppp/p4n2/1pp3B1/3pP3/PB1P4/1PP1NPPP/R2Q1RK1 b - - 1 12'::chess_board;
                            chess_board
------------------------------------------------------------------
 r2q1rk1/1b1pbppp/p4n2/1pp3B1/3pP3/PB1P4/1PP1NPPP/R2Q1RK1 b KQkq - 1 12
(1 row)
```

Data Type Validation: Chess Game

- Validate creating table with chess_game data type followed by insert and select statement

- Validate type cast operation from string to chess_game data type

```
postgres=# DROP TABLE IF EXISTS chessgame1;
DROP TABLE
postgres=# CREATE TABLE chessgame1(pgn chess_game);
CREATE TABLE
postgres=# INSERT INTO chessgame1
VALUES('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6');
INSERT 0 1
postgres=# SELECT * FROM chessgame1;
                    pgn
-----------------------------------------
 1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6
(1 row)
```

```
postgres=#
postgres=# SELECT '1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O b5'::chess_game;
                   chess_game
-----------------------------------------------
 1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O b5
(1 row)
```

Function Validation: getBoard

- Validate if **getBoard** function is returning correct value for static chess game and number of half moves being passed

```
postgres=# SELECT getBoard('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3',0);
                          getboard
-------------------------------------------------------------
 rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
(1 row)

postgres=# SELECT getBoard('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3',1);
                          getboard
-------------------------------------------------------------
 rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR b KQkq e3 0 1
(1 row)

postgres=# SELECT getBoard('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3',2);
                          getboard
-------------------------------------------------------------
 rnbqkbnr/pppp1ppp/8/4p3/4P3/8/PPPP1PPP/RNBQKBNR w KQkq e6 0 2
(1 row)
```

- Validate if **getBoard** function can be used to query table containing field with chess game data type

```
postgres=# SELECT getBoard(pgn,4) FROM chessgame limit 10;
                          getboard
-------------------------------------------------------------
 rnbqkbnr/p1ppppp1/1p5p/8/P6P/8/1PPPPPP1/RNBQKBNR w KQkq - 0 3
 rnbqkb1r/1ppppppp/p6n/8/6P1/2P5/PP1PPP1P/RNBQKBNR w KQkq - 1 3
 rnbqkbnr/pppppppp/8/8/6P1/8/PPPPPP1P/RNBQKBNR w KQkq - 1 3
 rnbqkbnr/1pppp1pp/p4p2/8/P7/2N5/1PPPPPPP/R1BQKBNR w KQkq - 0 3
 rnbqkbnr/1ppppp1p/p5p1/8/3P4/7N/PPP1PPPP/RNBQKB1R w KQkq - 0 3
 rnb1kbnr/ppppqppp/4p3/8/8/N5P1/PPPPPP1P/R1BQKBNR w KQkq - 1 3
 rnbqkbnr/p1ppppp1/7p/1p6/6P1/5P2/PPPPP2P/RNBQKBNR w KQkq - 0 3
 rnbqkbnr/1ppppp1p/p7/6p1/P7/5P2/1PPPP1PP/RNBQKBNR w KQkq - 0 3
 rnbq1bnr/pppppkpp/5p2/8/6P1/P7/1PPPPP1P/RNBQKBNR w KQ - 1 3
 rnbqkbnr/pp1ppp1p/8/2p3p1/5P2/3P4/PPP1P1PP/RNBQKBNR w KQkq g6 0 3
(10 rows)
```

Function Validation: getFirstMove

- Validate if **getFirstMove** function is returning correct value for static chess game and number of half moves being passed

- Validate if **getFirstMove** function can be used to query table containing field with chess game data type

```
postgres=# SELECT getFirstMoves('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3', 0);
 getfirstmoves
---------------

(1 row)

postgres=# SELECT getFirstMoves('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3', 1);
 getfirstmoves
---------------
 1. e4
(1 row)

postgres=# SELECT getFirstMoves('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3', 2);
 getfirstmoves
---------------
 1. e4 e5
(1 row)

postgres=# SELECT getFirstMoves('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3', 3);
  getfirstmoves
-----------------
 1. e4 e5 2. Nf3
(1 row)

postgres=# SELECT getFirstMoves('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3', 4);
    getfirstmoves
---------------------
 1. e4 e5 2. Nf3 Nc6
(1 row)
```

```
postgres=# SELECT getFirstMoves(pgn,3),pgn FROM chessgame limit 10;
```

```
                    pgn
----------------+----------------------------------------------------------------------------------------------------
 1. a4 h6 2. h4  | 1. a4 h6 2. h4 b6 3. Nf3 Rh7 4. b3 Nf6 5. Nc3 Bb7 6. d4 d5 7. a5 Nc6 8. Bg5 Qd6 9. Rh2 Rh8 10. e3 Nh5 11. Ra2 f5 12. a6 Nf4 13. Bc4 O-O-O
 14. Bxh6 Ba8 15. Ra3 Na5 16. Bd3 Nxb3 17. Na2 Nxd4 18. Nc3 e5 19. Qb1 Nfe2 20. Qd1 Nb3 21. Bf4 Nbc1 22. Bc4 Qc5 23. Ba2 Qa5 24. g4 Nb3 25. Rxa5 Nc5 26. Ne4
 exf4 27. Kf1 fxe3 28. Ned2 Rd6 29. Ra3 Kb8 30. Bc4 Be7 31. Qb1 Rh7 32. h5 g5 33. Qb2 Rd8 34. Ne1 Nd7 35. f3 Nf4 36. Qb1 b5 37. Rh4 exd2 38. Be2 d1=R 39. Ra
4 Rf7 40. Rxf4 Bc5 41. Qc1 Rf6 42. Rh2 Be3 43. Qa1 Rc8 44. Qa3 Bc5 45. Qc1 Be7 46. Re4 Rc6 47. Qb1 d4 48. Qa1 Bf8 49. Re7 Nf6 50. Qc3 f4 51. Re6 Be7 52. Rg2
 Rf8 53. Qa3 Rb6 54. Qb4 Rd2 55. Rf2 Rg8 56. Rxf6 Rxa6 57. Qa4 Rg7 58. Bxb5 Ba3 59. Bd3 Rxc2 60. Rf5 Rf6 61. Qa5 Rfc6 62. Qd2 R2c5 63. Bc2 Ra5 64. Qc3 Rf7 6
5. Qa1 Kc8 66. Be4 R6c5 67. Rf6 Rxf6 68. Bc6 Rf7 69. Bb7+ Kd8 70. Bc6 Bb4 71. Ke2 Re5+ 72. Kd1 Re2 73. Bd7 Kxd7 74. Qa3 Bd5 75. Qa2 Ba8 76. h6 Rb2 77. Qb1 K
e6 78. Qc1 Kf6 79. Rxb2 Bf8 80. Qe3 Kg6 81. Rd2 Bxh6 82. Rd3 c6 83. Qe5 Rf5 84. Qb5 Kf7 85. Rd2 Kg6 86. Qa4 Bb7 87. Qa1 Ra4 88. Rc2 Rb4 89. Ke2 a6 90. Rb2 B
a8 91. Ra2 Rb6 92. Ra3 d3+ 93. Rxd3 Rc5 94. Qd1 Bb7 95. Qa4 Kf7 96. Qxa6 Rc4 97. Rd8 R6b4 98. Kd2 Kg7 99. Ke2 Rc1 100. Qb5 Rb2+ 101. Nc2 R2b1 102. Qd3 Rb3 1
03. Rd5 Kf7 104. Qh7+ Kf8 105. Ne3 Rbc3 106. Rc5 Rd3 107. Nf5 Rc4 108. Qe7+ Kg8 109. Rxc6 Rd7 110. Ng3 R7d4 111. Qf8+ Kh7 112. Rc8 Be4 113. Nh1 Rc6 114. Qb4
 Rcd6 115. Ke1 Bc6 116. Rg8 R6d5 117. Qc4 Rd3 118. Rd8 Rb3 119. Qxc6 Rb6 120. Ke2 R6b5 121. Rxd5 Rb1 122. Qc7+ Kg8 123. Rc5 Kh8 124. Rb5 Kg8 125. Qc3 Rb4 12
6. Qc6 Rd4 127. Qe8+ Kg7 128. Qc8 Rd3
 1. g4 a6 2. c3  | 1. g4 a6 2. c3 Nh6 3. h4 e5 4. Nf3 a5 5. b4 Nc6 6. g5 Ne7 7. bxa5 Rg8 8. g6 b5 9. Ba3 Ba6 10. Bd6 Qb8 11. Nd4 Ra7 12. Bb4 hxg6 13. Qc1 Ne
f5 14. Qa3 d5 15. e4 Nxd4 16. Bc5 Bb7 17. Be7 Ra8 18. d3 g5 19. Qc1 g4 20. Na3 g5 21. Rb1 Qd8 22. Nxb5 Rb8 23. Na7 Bxe7 24. Kd2 Ba3 25. Rg1 c6 26. Rxg4 Bd6
27. Bg2 Bc7 28. Rb2 Qf6 29. Rg3 Qe7 30. Rb4 Ndf5 31. Rb3 Rd8 32. Qd1 Bc8 33. a6 Bb7 34. Rxg5 Qa3 35. Qa1 Qc5 36. Rxg8+ Kd7 37. Qg1 Qc4 38. Ra3 Ke7 39. Ra5 f
6 40. Rb5 Ne3 41. Bh3 dxe4 42. Kc1 Qxc3+ 43. Kb1 Neg4 44. Bf1 Nf7 45. Re8+ Kd6 46. dxe4 Ra8 47. axb7 f5 48. Rg8 c5 49. b8=R Qg3 50. Rge8 Qxf2 51. a4 Qf3 52.
 R8b7 Rxa7 53. R5b6+ Bxb6 54. Qxg4 Rxa4 55. Qxf3 Nh8 56. Ba6 Ra3 57. Qg2 Rc3 58. Bd3 Kc6 59. h5 f4 60. Ka2 Bc7 61. Rb2 f3 62. Qd2 Rc4 63. Rb1 Ng6 64. Rf1 Ra
4+ 65. Kb1 Bb8 66. Rxb8 Ra6 67. Kc2 Ne7 68. Ra1 Kd6 69. Qc1 c4 70. Rc8 Ng6 71. Qe1 Ra8 72. Qc1 Ra4 73. Kd1 Nf4 74. Kd2 Ng6 75. Rd8+ Kc6 76. Qf1 Nh4 77. Rxa4
```

Function Validation: hasOpening

- Validate if **hasOpening** function is returning correct value for static chess games passed.

- Validate if **hasOpening** function can be used to query table containing field with chess game data type.

```
postgres=# SELECT hasOpening('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3', '1. e4 e5 2. Nf3 Nc6');
 hasopening
------------
 t
(1 row)

postgres=# SELECT hasOpening('1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. d3', '1. Nf3 c6 2. a3 d6');
 hasopening
------------
 f
(1 row)
```

```
postgres=# SELECT * FROM chessgame WHERE hasOpening(pgn,'1. e4 d6') LIMIT 10;
```

```
          pgn
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
--------------------------------
 1. e4 d6 2. g4 Nc6 3. Ke2 g6 4. Ke3 Bd7 5. a4 Bc8 6. Kf4 e6 7. Be2 Bh6+ 8. g5 Na5 9. Nf3 Kf8 10. Ne1 Bd7 11. gxh6 Bc6 12. Nc3 e5+ 13. Kg4 Nf6+ 14. Kg3 Ke7
15. Bd3 d5 16. b3 Ne8 17. Bc4 dxe4 18. Nf3 Rc8 19. Nb5 f5 20. c3 Qd6 21. Qe2 Qe6 22. Rf1 Bd7 23. Re1 c5 24. Ra3 f4+ 25. Kh4 Kf8 26. Nbd4 Qc6 27. Ra2 Nc7 28.
 Bf7 Ke7 29. Qc4 Bh3 30. Nxe5 Qe8 31. Rxe4 Qg8 32. Be6 b6 33. Qb5 Na8 34. Nf7 Bf5 35. Bxf5+ Kf8 36. Qxc5+ Rxc5 37. Re2 gxf5 38. Re3 Re5 39. Ng5 Re6 40. Bb2
Ke7 41. Ne2 Kf8 42. Re5 Rc6 43. d3 b5 44. d4 Qg6 45. Nxf4 Nb7 46. b4 Qg8 47. Bc1 Nc7 48. Ng2 a5 49. Rxf5+ Rf6 50. f3 Qc4 51. Kh5 Ke8 52. Re2+ Kd7 53. Nf4 ax
b4 54. axb5 Na6 55. Rc2 Qe6 56. d5 bxc3 57. Re2 Qe7 58. d6 Ke8 59. Rb2 Qe5 60. Rb1 Nb8 61. Nd3 Rxd6 62. Nxh7 Qd4 63. h3 c2 64. Rg5 Rg6 65. Nb4 Qg1 66. Rd5 R
f6 67. Na2 Qh1 68. Rd3 Nc5 69. Nb4 Kf7 70. Ng5+ Ke7 71. Ra1 Rh7 72. Ra2 Qxc1 73. Rd2 Na4 74. Nd3 Ke8 75. Nf4 Rd7 76. Ra3 Rc6 77. Ra2 Nb2 78. bxc6 Na6 79. Ng
e6 Qb1 80. Rxb2 Qa2 81. Rbxc2 Rxd2 82. c7 Rd1 83. Rb2 Rd4 84. Kh4 Ra4 85. c8=N Nc5 86. Rxa2 Nd3 87. Ra3 Rd4 88. Kg4 Re4 89. Rb3 Rb4 90. Kh4 Rb6 91. h7 Rb7 9
2. Rxb7 Ne5 93. h8=Q#
 1. e4 d6 2. c3 g5 3. Qa4+ b5 4. Qa5 g4 5. Qxa7 c6 6. d4 Be6 7. h4 h6 8. f3 Qc7 9. g3 Qd8 10. Kd2 Nd7 11. Rh3 Rh7 12. Bd3 gxh3 13. Kc2 Rb8 14. Kd1 b4 15. Qx
b8 Nc5 16. a4 Rg7 17. Be2 Bd7 18. Bd2 Rxg3 19. Bd3 Qxb8 20. c4 Rg6 21. d5 Ne6 22. Ke2 Nc5 23. Na3 Qb7 24. Bf4 Rf6 25. Ra2 Na6 26. e5 c5 27. Bg3 e6 28. Ra1 b
xa3 29. Bf4 Qxb2+ 30. Bc2 Nb4 31. Ke1 Bb5 32. Be4 Ne7 33. Rxa3 Rxf4 34. Bh7 Qc3+ 35. Kf2 f6 36. a5 Qxf3+ 37. Ke1 Qe4+ 38. Kd2 Qg2+ 39. Ke3 Ng6 40. Nf3 Na6 4
1. Nh2 Qc2 42. Rd3 Qxd3+ 43. Kxd3 Be7 44. exd6 h5 45. Kd2 Nb4 46. Ke2 Kf8 47. dxe7+ Kxe7 48. Ke1 Kd6 49. Bg8 Na2 50. Ng4 Bc6 51. Ne3 Rxc4 52. Ke2 Ne7 53. Ke
1 Rc3 54. Nf5+ Ke5 55. Nxe7 Be8 56. Ke2 Bc6 57. Bf7 Rc2+ 58. Kd1 Bd7 59. Bg8 Ke4 60. Nc8 Bxc8 61. Bxe6 f5 62. Bd7 Kd3 63. Ba4 f4 64. Bd7 h2 65. Be6 Rf2 66.
Bf5+ Kc4 67. Bh7 Rd2+ 68. Kxd2 Kb5 69. Bb1 Bb7 70. a6 Kc4 71. Bd3+ Kd4 72. Bf5 Bc8 73. a7 Kxd5 74. Be6+ Kd4 75. Bxa2 Bb7 76. a8=N Bh1 77. Nb6 c4 78. Bb1 Bb7
 79. Nd5 h1=R 80. Be4 Kxe4 81. Nc3+ Kd4 82. Ne2+ Kd5 83. Kc2 c3 84. Nc1 Ba6 85. Nb3 Bd3+ 86. Kxd3 Ke5 87. Kxc3 Ra1 88. Nc5 Kd5 89. Nb3 Kc6 90. Nxa1 Kc5 91.
Nb3+ Kb6 92. Kc2 Kc6 93. Nc1 Kd7 94. Kd2 f3 95. Ke3 Ke8 96. Kd4 Kf8 97. Kc4 Kg8 98. Nb3 Kg7 99. Nd2 f2 100. Kb3 f1=R 101. Nxf1 Kf7 102. Ka2 Kg7 103. Ne3 Kh6
 104. Kb1 Kh7 105. Kb2 Kg8 106. Ka2 Kf7 107. Nc4 Kg7 108. Ne3 Kf7 109. Nf1 Ke6 110. Ne3 Kf6 111. Ng2 Ke7 112. Nf4 Kd6 113. Ka3 Ke5 114. Ng2 Ke6 115. Ka2 Ke7
```

```
postgres=# SELECT count(*) FROM chessgame WHERE hasOpening(pgn,'1. e4 d6');
 count
-------
    27
(1 row)
```

Function Validation: hasBoard

- Validate if **hasBoard** function is returning correct value for static chess game, chess board and number of half moves passed
- Validate if **hasBoard** function can be used to query table containing field with chess game data type
- Confirm that only chess board state should be considered

```
postgres=# SELECT
hasBoard(
    '1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O b5 6. Bb3 Bb7 7. d3 Be7 8. Nc3 O-O 9. a3 Nd4 10. Nxd4 exd4',
    'r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R b KQkq - 3 3',
    1
);
 hasboard
----------
 f
(1 row)

postgres=# SELECT
hasBoard(
    '1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O b5 6. Bb3 Bb7 7. d3 Be7 8. Nc3 O-O 9. a3 Nd4 10. Nxd4 exd4',
    'r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R b KQkq - 3 3',
    2
);
 hasboard
----------
 f
(1 row)

postgres=# SELECT
hasBoard(
    '1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O b5 6. Bb3 Bb7 7. d3 Be7 8. Nc3 O-O 9. a3 Nd4 10. Nxd4 exd4',
    'r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R b KQkq - 3 3',
    3
);
 hasboard
----------
 f
postgres=# SELECT
hasBoard(
    '1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O b5 6. Bb3 Bb7 7. d3 Be7 8. Nc3 O-O 9. a3 Nd4 10. Nxd4 exd4',
    'r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R b KQkq - 3 3',
    4
);
 hasboard
----------
 f
(1 row)

postgres=# SELECT
hasBoard(
    '1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O b5 6. Bb3 Bb7 7. d3 Be7 8. Nc3 O-O 9. a3 Nd4 10. Nxd4 exd4',
    'r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R b KQkq - 3 3',
    5
);
 hasboard
----------
 t
(1 row)

postgres=# SELECT
hasBoard(
    '1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O b5 6. Bb3 Bb7 7. d3 Be7 8. Nc3 O-O 9. a3 Nd4 10. Nxd4 exd4',
    'r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R b KQkq - 3 3',
    6
);
 hasboard
----------
 t
(1 row)
```

```
postgres=# SELECT * FROM chessgame WHERE hasBoard(pgn,'rnbqkbnr/pppp1ppp/8/4p3/4P3/8/PPPP1PPP/RNBQKBNR w KQkq e6 0 2',5);
```

```
              pgn
--------------------------------------------------------------------------------------------------------------
 1. e3 e5 2. e4 d5 3. b4 Nh6 4. Be2 c5 5. Bf3 Qd7 6. Nh3 Ke7 7. Ng1 Qc7 8. Ne2 Nf5 9. Nbc3 Be6 10. b5 Qc6 11. d4 Qa6 12. g4 Nh4 13. h3 Qb6 14. Qd2 Kd7 15. Q
f4 g5 16. Qf5 h5 17. Na4 dxe4 18. d5 Qd8 19. Nd4 Qc8 20. Bd2 Kc7 21. Rb1 Bd7 22. Be3 Kd8 23. Qf4 Nc6 24. gxh5 Nxd4 25. Bd1 g4 26. Qf3 Nb3 27. hxg4 Qc7 28. N
xc5 Na5 29. Qe2 Qb6 30. Bf4 Bxg4 31. Nxb7+ Ke8 32. Ra1 Bxe2 33. Rxh4 Rh7 34. Kxe2 Rb8 35. Rh1 Qc7 36. a3 Nc4 37. Bg5 Qa5 38. a4 Bb4 39. Rh4 f6 40. f4 Nd2 41
. h6 Nb3 42. Nxa5 e3 43. Nb7 Be7 44. Rh3 Nc1+ 45. Ke1 Rh8 46. Ra3 Rxh6 47. Rh1 Bb4+ 48. Rc3 Kd7 49. Kf1 exf4 50. Kg1 Rh7 51. Bh6 Rd8 52. Nc5+ Kd6 53. Nb3 Nd
3 54. Na5 Nc5 55. Bg4 Na6 56. Bd1 f3 57. Ra3 Nc5 58. Rxe3 Ra8 59. Ra3 Bxa3 60. Rh4 Nb7 61. Rh2 Re7 62. Nc4+ Kxd5 63. Ne3+ Ke5 64. Nf5 Rd8 65. Bf8 Rb8 66. Ne
3 Kd4 67. Nd5 Re5 68. Kf2 Kc4 69. Nf4 Bd6 70. c3 Be7 71. Rh1 Na5 72. Nh5 Rxh5 73. Bh6 Kd3 74. Rh4 a6 75. b6 Nc6 76. Bg7 Na7 77. Bxf3 Rb7 78. Bg2 Rb5 79. Be4
+ Kd2 80. c4 Bd6 81. c5 a5 82. Rh6 Kc1 83. Ke2 Bf4 84. Rh7 Rb8 85. Ke1 R8xb6 86. Kf1 Rb4 87. Rh1 Rd6 88. Rh5 Nc6 89. Kg1 Rd7 90. Rh1 Kb2 91. Kg2 Bc7 92. Bh8
 Nd8 93. Rd1 Ka2 94. Rh1 Rd1 95. Bb1+ Rdxb1 96. Kf3 Bb8 97. Rxb1 Rb2 98. Kg4 Rb4+ 99. Kh5 Rh4+ 100. Kxh4 Ne6 101. Rxb8 Ng7 102. Rf8 Kb2 103. Rxf6 Kc3 104. R
b6 Kc2 105. Rg6 Kb2 106. Rf6 Kb3 107. Bxg7 Kc4 108. Bf8 Kb4 109. Rf4+ Kb3 110. Rc4 Ka3 111. Be7 Ka2 112. Rb4 axb4 113. Kg4 Kb1 114. Kh5 Ka1 115. Kh6 Ka2 116
. Bd6 Kb1 117. Bg3 Kc1 118. Be5 Kd2 119. Bg3 Ke3 120. Bb8 Kf2 121. Kh7 Kf1 122. Bh2 Kf2 123. Kg7 Ke2 124. Bb8 Kd2 125. Kh6 Kc3 126. Be5+ Kb3 127. Kg7 Kxa4 1
28. Kf6 Kb5
 1. e4 e5 2. Bd3 b5 3. h4 f5 4. Qe2 a5 5. a3 Na6 6. Qg4 Bb4 7. Qxg7 Nc5 8. Bf1 Ra6 9. Qh6 Rf6 10. Ne2 Rc6 11. Qe6+ Kf8 12. Qe7+ Qxe7 13. Rg1 Na6 14. g4 Rd6
15. Bh3 Qxh4 16. Ng3 fxg4 17. Nh5 Bxd2+ 18. Kd1 Bc3+ 19. Nd2 Rf6 20. b4 Qg5 21. f4 Bb2 22. Bxg4 a4 23. f5 Qg7 24. Bf3 Nc5 25. Be2 Qh6 26. Ke1 Bxa1 27. bxc5
b4 28. Bg4 Ba6 29. Bd1 Bc3 30. Rg5 Rxf5 31. c6 Qxg5 32. Bb2 Bxd2#
 1. e4 e5 2. Bb5 f6 3. Qf3 g5 4. Qe2 Kf7 5. Qg4 d6 6. Bf1 b5 7. c4 b4 8. d3 Bh6 9. Qe6+ Kg6 10. Qe7 Nc6 11. Kd2 Bg7 12. Kc2 Qd7 13. Qe6 Na5 14. Bxg5 Qb5 15.
 Be2 Ba6 16. Be3 Nb3 17. Qe7 Rb8 18. Kd1 Ra8 19. f4 Kh6 20. f5#
 1. e4 e5 2. Qe2 Bb4 3. Kd1 Nf6 4. d3 h5 5. Nd2 Nxe4 6. c4 Be7 7. Qg4 f6 8. Rb1 a6 9. Qh3 Ng3 10. fxg3 Ra7 11. Ra1 Bd6 12. Qxh5+ Ke7 13. Qxh8 Bc5 14. d4 Qg8
:
```

```
postgres=# SELECT count(*) FROM chessgame WHERE hasBoard(pgn,'rnbqkbnr/pppp1ppp/8/4p3/4P3/8/PPPP1PPP/RNBQKBNR w KQkq e6 0 2',5);
 count
-------
    33
(1 row)
```

```
postgres=# SELECT hasboard('1. e4 e5
2. Nf3 Nc6
3. Bb5 a6
4. Ba4 Nf6
5. d3'::chess_game,'rnbqkbnr/pppp1ppp/8/4p3/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 0 2'::chess_board,4);
 hasboard
----------
 t
(1 row)

postgres=# SELECT hasboard('1. e4 e5
2. Nf3 Nc6
3. Bb5 a6
4. Ba4 Nf6
5. d3'::chess_game,'rnbqkbnr/pppp1ppp/8/4p3/4P3/8/PPPP1PPP/RNBQKBNR w KQkq e6 0 2'::chess_board,4);
 hasboard
----------
 t
(1 row)
```

Index Validation: B-Tree

- Validate if we are able to create **B-Tree** index on chess_game datatype

- Validate if **B-Tree** index is being picked by **hasOpening** function

```
postgres=# DROP TABLE IF EXISTS chessgame;
DROP TABLE
postgres=# CREATE TABLE chessgame(pgn chess_game);
CREATE TABLE
postgres=# COPY chessgame(pgn)
FROM '/mnt/c/ULB/Database Systems Architecture/Project/git/DSA_ChessGame/sample_pgn10000.csv'
DELIMITER ','
CSV HEADER;
COPY 10000
postgres=# EXPLAIN ANALYZE SELECT * FROM chessgame WHERE hasOpening(pgn,'1. e3 b5');
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Seq Scan on chessgame  (cost=0.00..817.08 rows=2501 width=512) (actual time=689.440..11197.379 rows=28 loops=1)
   Filter: ((pgn >= '1. e3 b5'::chess_game) AND (pgn < '1. e3 b5'::chess_game))
   Rows Removed by Filter: 9972
 Planning Time: 0.102 ms
 Execution Time: 11197.420 ms
(5 rows)

postgres=# CREATE INDEX idx ON chessgame(pgn);
CREATE INDEX
postgres=# SET enable_seqscan=OFF;
SET
postgres=# EXPLAIN ANALYZE SELECT * FROM chessgame WHERE hasOpening(pgn,'1. e3 b5');
                                                            QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using idx on chessgame  (cost=0.54..906.53 rows=2500 width=512) (actual time=19.503..77.812 rows=28 loops=1)
   Index Cond: ((pgn >= '1. e3 b5'::chess_game) AND (pgn < '1. e3 b5'::chess_game))
   Heap Fetches: 0
 Planning Time: 0.760 ms
 Execution Time: 77.939 ms
(5 rows)
```

Index Validation: GIN

- Validate if we are able to create **GIN** index on chess_game datatype
- Validate if **GIN** index is being picked by **hasBoard** function

```
postgres=# DROP TABLE IF EXISTS chessgame;
DROP TABLE
postgres=# CREATE TABLE chessgame(pgn chess_game);
CREATE TABLE
postgres=# COPY chessgame(pgn)
FROM '/mnt/c/ULB/Database Systems Architecture/Project/git/DSA_ChessGame/sample_pgn10000.csv'
DELIMITER ','
CSV HEADER;
COPY 10000
postgres=# EXPLAIN ANALYZE SELECT * FROM chessgame WHERE hasBoard(pgn,'rnbqkbnr/pppp1ppp/8/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq - 1 2',10);
                                                                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on chessgame  (cost=0.00..817.08 rows=5002 width=512) (actual time=601.504..8065.450 rows=3 loops=1)
   Filter: (pgn @@ concat('10'::character varying, ',', 'rnbqkbnr/pppp1ppp/8/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq - 1 2'::character varying))
   Rows Removed by Filter: 9997
 Planning Time: 0.262 ms
 Execution Time: 8065.474 ms
(5 rows)

postgres=# CREATE INDEX idx ON chessgame using gin(pgn);
CREATE INDEX
postgres=# EXPLAIN ANALYZE SELECT * FROM chessgame WHERE hasBoard(pgn,'rnbqkbnr/pppp1ppp/8/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq - 1 2',10);
                                                                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on chessgame  (cost=66.75..808.75 rows=5000 width=512) (actual time=0.041..0.064 rows=3 loops=1)
   Recheck Cond: (pgn @@ concat('10'::character varying, ',', 'rnbqkbnr/pppp1ppp/8/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq - 1 2'::character varying))
   Heap Blocks: exact=3
   ->  Bitmap Index Scan on idx  (cost=0.00..65.50 rows=5000 width=0) (actual time=0.035..0.036 rows=3 loops=1)
         Index Cond: (pgn @@ concat('10'::character varying, ',', 'rnbqkbnr/pppp1ppp/8/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq - 1 2'::character varying))
 Planning Time: 1.407 ms
 Execution Time: 0.096 ms
(7 rows)
```

```
postgres=# EXPLAIN ANALYZE SELECT * FROM chessgame WHERE hasBoard(pgn,'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1',10);
                                                                QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on chessgame  (cost=66.75..808.75 rows=5000 width=512) (actual time=0.726..3.026 rows=10000 loops=1)
   Recheck Cond: (pgn @@ concat('10'::character varying, ',', 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1'::character varying))
   Heap Blocks: exact=667
   ->  Bitmap Index Scan on idx  (cost=0.00..65.50 rows=5000 width=0) (actual time=0.668..0.668 rows=10000 loops=1)
         Index Cond: (pgn @@ concat('10'::character varying, ',', 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1'::character varying))
 Planning Time: 0.147 ms
 Execution Time: 3.317 ms
(7 rows)
```

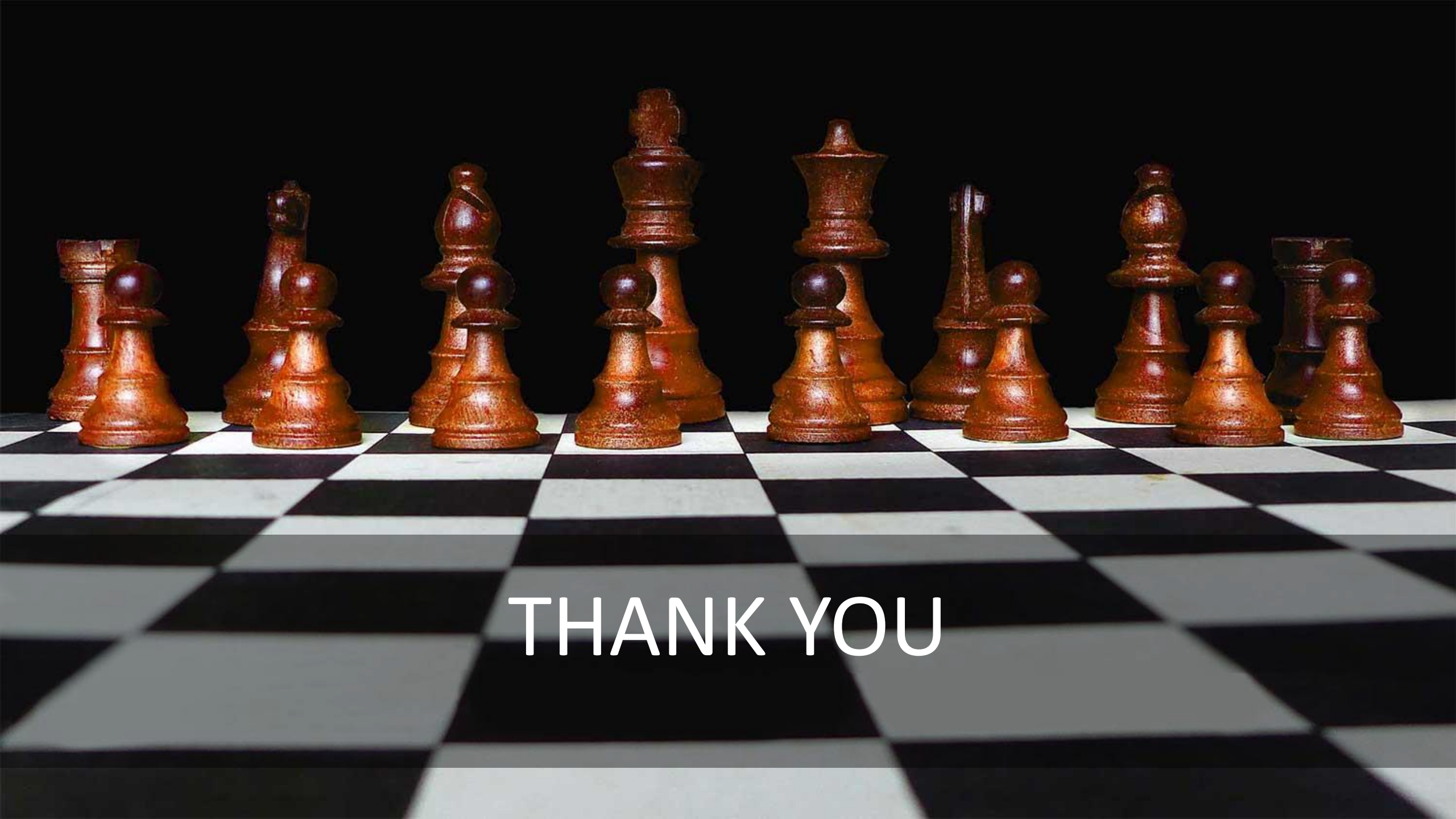
Important Links

Github

https://github.com/SonyShrestha/DSA_ChessGame

Test Cases

https://github.com/SonyShrestha/DSA_ChessGame/blob/main/sql_files/test_script.sql


THANK YOU