競馬データ分析サイト（racescore-web）運用マニュアル

バージョン: 1.2
最終更新日: 2025年12月20日
作成者: Manus AI

目次

---

1. システム概要

本システムは、SmartRC（https://www.smartrc.jp/v3/）を参考に開発された競馬データ分析サイトです。レースカードに各種指数（L4F、T2F、ポテンシャル、レボウマ、巻き返し、クッション値）を表示し、過去走データと統合して馬の能力を多角的に評価することを目的としています。

主な機能

機能	説明
レースカード表示	出走馬一覧と各種指数を表示
過去走データ表示	各馬の直近5走のデータと指数を表示
競うスコア計算	巻き返し指数ベースの独自スコアリング
指数CSVアップロード	6種類の指数データを一括アップロード
クラスタタイム分析	同日±1日の別クラスタタイムとの比較

---

2. システム要件

サーバー環境

項目	要件
OS	Ubuntu 22.04 LTS（推奨）/ Windows 10以降
Node.js	v18.0.0 以上
npm/pnpm	npm 9.x 以上 / pnpm 8.x 以上
データベース	SQLite 3.x（better-sqlite3）

クライアント環境（CSVアップロード用）

項目	要件
OS	Windows $^{10}/_{11}$
Node.js	v18.0.0 以上
必要なnpmパッケージ	ts-node, typescript, papaparse

---

3. ディレクトリ構成

```
racescore-web/
├── app/                          # Next.js App Router
│   ├── components/               # UIコンポーネント
│   │   ├── DateSelector.tsx      # 日付選択
│   │   ├── EntryTable.tsx        # 出馬表
│   │   ├── RaceCard.tsx          # レースカード
│   │   └── UploadForm.tsx        # アップロードフォーム
│   ├── race/[raceKey]/page.tsx   # レース詳細ページ
│   ├── races/[ymd]/page.tsx      # 日付別レース一覧
│   └── page.tsx                  # トップページ
│
├── pages/api/                    # API Routes
│   ├── race-card-with-score.ts   # レースカード+スコアAPI
│   ├── race-card.ts              # レースカードAPI
│   ├── races.ts                  # レース一覧API
│   └── upload-indices.ts         # 指数アップロードAPI
│
├── tools/                        # ツール類
│   └── upload-indices.ts         # 指数CSVマージ＆アップロード
│
├── utils/                        # ユーティリティ
│   └── getClusterData.ts         # クラスタデータ・スコア計算
│
├── lib/                          # ライブラリ
│   └── db-new.ts                 # データベース接続
│
├── types/                        # 型定義
│   └── record.ts                 # RecordRow型
│
├── sync-indices.bat              # Windows用アップロードバッチ
├── races.db                      # SQLiteデータベース
└── package.json                  # 依存関係
```

---

4. データベース構造

本システムはSQLiteデータベース（`races.db`）を使用しています。

4.1 indicesテーブル（指数データ）

各馬の各レースにおける指数を格納するテーブルです。

```
CREATE TABLE indices (
  race_id TEXT PRIMARY KEY,     -- 18桁のレースID（16桁レースID + 2桁馬番）
  L4F REAL,                     -- 後半4ハロン指数
  T2F REAL,                     -- 前半2ハロン指数
  potential REAL,               -- ポテンシャル指数
  revouma REAL,                 -- レボウマ指数
  makikaeshi REAL,              -- 巻き返し指数
  cushion REAL,                 -- クッション値
  created_at TEXT,              -- 作成日時
  updated_at TEXT               -- 更新日時
);
```

race_idの構成:

位置	桁数	内容	例
1-8	8桁	日付（YYYYMMDD）	20250504
9-10	2桁	競馬場コード	05（東京）
11-12	2桁	開催回	05
13-14	2桁	開催日	01
15-16	2桁	レース番号	12
17-18	2桁	馬番	08

競馬場コード一覧:

コード	競馬場	コード	競馬場
01	札幌	06	中山
02	函館	07	中京
03	福島	08	京都
04	新潟	09	阪神
05	東京	10	小倉

4.2 umadataテーブル（過去走データ）

各馬の過去走成績を格納するテーブルです。

```
CREATE TABLE umadata (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  race_id_new_no_horse_num TEXT,  -- レースID（馬番なし16桁）
  date TEXT,                      -- 日付
  distance TEXT,                  -- 距離（例: "芝1600"）
  horse_number TEXT,              -- 馬番
  horse_name TEXT,                -- 馬名
  index_value TEXT,               -- 指数値
  class_name TEXT,                -- クラス名
  track_condition TEXT,           -- 馬場状態
  finish_position TEXT,           -- 着順
  last_3f TEXT,                   -- 上がり3F
  finish_time TEXT,               -- 走破タイム
  pci TEXT,                       -- PCI（ペース指数）
  corner_2 TEXT,                  -- 2コーナー通過順
  corner_3 TEXT,                  -- 3コーナー通過順
  corner_4 TEXT,                  -- 4コーナー通過順
  jockey TEXT,                    -- 騎手
  trainer TEXT,                   -- 調教師
  place TEXT,                     -- 競馬場
  number_of_horses TEXT,          -- 出走頭数
  popularity TEXT,                -- 人気
  margin TEXT,                    -- 着差
  -- その他多数のカラム
  created_at TEXT
);
```

4.3 wakujunテーブル（出走表）

当日の出走馬情報を格納するテーブルです。

```
CREATE TABLE wakujun (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  date TEXT,                -- 日付（MMDD形式）
  place TEXT,               -- 競馬場名
  race_number TEXT,         -- レース番号
  class_name_1 TEXT,        -- クラス名1
  class_name_2 TEXT,        -- クラス名2
  waku TEXT,                -- 枠番
  umaban TEXT,              -- 馬番
  kinryo TEXT,              -- 斤量
  umamei TEXT,              -- 馬名
  seibetsu TEXT,            -- 性別
  nenrei TEXT,              -- 年齢
  kishu TEXT,               -- 騎手
  track_type TEXT,          -- コース種別（芝/ダ）
  distance TEXT,            -- 距離
  tosu TEXT,                -- 出走頭数
  shozoku TEXT,             -- 所属
  chokyoshi TEXT,           -- 調教師
  created_at TEXT
);
```

---

5. 指数の種類と説明

本システムでは6種類の指数を使用しています。

5.1 L4F（後半4ハロン指数）

項目	内容
概要	レース後半4ハロン（約800m）の走破能力を数値化
範囲	0〜100（高いほど良い）
用途	末脚の強さ、追い込み馬の評価
データソース	`C:\競馬データ\L4F\2025\*.csv`

5.2 T2F（前半2ハロン指数）

項目	内容
概要	レース前半2ハロン（約400m）の走破能力を数値化
範囲	0〜100（高いほど良い）
用途	スタートダッシュ力、逃げ馬の評価
データソース	`C:\競馬データ\T2F\2025\*.csv`

5.3 ポテンシャル指数

項目	内容
概要	馬の潜在能力を総合的に評価した指数
範囲	0〜100（高いほど良い）
用途	能力上位馬の特定、格上挑戦馬の評価
データソース	`C:\競馬データ\ポテンシャル指数\2025\*.csv`

5.4 レボウマ指数

項目	内容
概要	革命的な走りを見せる可能性を評価
範囲	0〜10（高いほど変化の可能性大）
用途	穴馬発掘、激走候補の特定
データソース	`C:\競馬データ\レボウマ\2025\*.csv`

5.5 巻き返し指数

項目	内容
概要	前走からの巻き返し可能性を評価
範囲	0〜10（高いほど巻き返し期待大）
用途	前走凡走馬の復活予測、競うスコア計算のベース
データソース	`C:\競馬データ\巻き返し指数\2025\*.csv`

5.6 クッション値

項目	内容
概要	馬場のクッション性（硬さ）を数値化
範囲	7.0〜12.0程度（値が大きいほど柔らかい）
用途	馬場適性の判断、重馬場巧者の特定
データソース	`C:\競馬データ\クッション値\2025\*.csv`

---

6. 競うスコアの計算式

競うスコアは、各馬の期待値を0〜100点で表す独自指標です。

6.1 計算式の概要

```
競うスコア = 巻き返し指数スコア（65点）
           + 着順スコア（10点）
           + 着差スコア（10点）
           + クラスタタイムスコア（8点）
           + 通過順位×ペーススコア（7点）
```

6.2 各要素の詳細

巻き返し指数スコア（最大65点）：

```
const comebackScore =
  (comeback1 / 10) * 50 +  // 前走: 50点満点
  (comeback2 / 10) * 10 +  // 2走前: 10点満点
  (comeback3 / 10) * 5;    // 3走前: 5点満点
```

着順スコア（最大10点）：

```
const finishScore = Math.max(0, 10 - (着順 - 1) * 1);
// 1着=10点, 2着=9点, ..., 10着以下=0点
```

着差スコア（最大10点）：

```
const marginScoreVal = Math.max(0, 10 - 着差秒 * 3);
// 0秒差=10点, 1秒差=7点, 3秒差以上=0点
```

通過順位×ペーススコア（最大7点）：

```
const basePassScore = (頭数 - 平均通過順位 + 1) / 頭数;
const passScore = basePassScore * ペース係数 * 7;
```

ペース係数:

ペース	係数
超ハイ	1.2
ハイ	1.1
ミドル	1.0
スロー	0.9
超スロー	0.8

6.3 印の自動割り当て

競うスコアに基づいて自動的に印を割り当てます。

順位	印
1位	◎
2位	○
3位	▲
4位	☆
5位	△

---

7. CSVアップロード手順

7.1 事前準備

1. **フォルダ構成の確認**

以下のフォルダ構成でCSVファイルを配置してください：

```
C:\競馬データ\
├── L4F\2025\
│   ├── 0504東京.csv
│   ├── 0504京都.csv
│   └── ...
├── T2F\2025\
│   └── ...
├── ポテンシャル指数\2025\
│   └── ...
├── レボウマ\2025\
│   └── ...
├── 巻き返し指数\2025\
│   └── ...
└── クッション値\2025\
    └── ...
```

1. **CSVファイル形式**

各CSVファイルはヘッダーなしの2列形式です：

```
 202505040501001,85.5
202505040501002,78.2
202505040501003,92.1
```

- 1列目: race_id（18桁）
- 2列目: 指数値（数値）

7.2 アップロード実行

方法1: バッチファイル実行（推奨）

1. プロジェクトルートの `sync-indices.bat` をダブルクリック
2. 処理完了を待つ（数分かかる場合あり）
3. 「SUCCESS! XXXXX records saved」と表示されれば成功

方法2: コマンドライン実行

```
cd /path/to/racescore-web
npx ts-node tools/upload-indices.ts
```

7.3 アップロード処理の流れ

1. 6つの指数フォルダからCSVファイルを読み込み
   ↓
2. race_idをキーに横方向マージ
   ↓
3. マージ結果をCSVファイルとして保存（ダウンロードフォルダ）
   ↓
4. APIエンドポイント（/api/upload-indices）にPOST
   ↓
5. SQLiteデータベースのindicesテーブルにUPSERT

7.4 アップロード後の確認

```
# SQLiteで直接確認
sqlite3 races.db "SELECT COUNT(*) FROM indices;"
sqlite3 races.db "SELECT * FROM indices LIMIT 5;"
```

---

8. APIエンドポイント

8.1 指数アップロードAPI

エンドポイント: `POST /api/upload-indices`

リクエストボディ:

```
{
  "data": [
    {
      "race_id": "202505040501001",
      "L4F": 85.5,
      "T2F": 72.3,
      "potential": 88.0,
      "revouma": 6.5,
      "makikaeshi": 7.2,
      "cushion": 9.8
    }
  ]
}
```


レスポンス:


```
{
  "success": true,
  "message": "46086件の指数データを保存しました",
  "count": 46086
}
```


8.2 レースカードAPI

エンドポイント: `GET /api/race-card-with-score`

パラメータ:

パラメータ	説明	例
date	日付（MMDD形式）	0504
place	競馬場名	東京
raceNumber	レース番号	12

レスポンス:

```
{
  "raceInfo": {
    "date": "0504",
    "place": "東京",
    "raceNumber": "12",
    "className": "3歳以上1勝クラス",
    "trackType": "芝",
    "distance": "1600",
    "fieldSize": 16
  },
  "horses": [
    {
      "umaban": "1",
      "umamei": "サンプルホース",
      "score": 78.5,
      "indices": {
        "L4F": 85.5,
        "T2F": 72.3,
        "potential": 88.0,
        "revouma": 6.5,
        "makikaeshi": 7.2,
        "cushion": 9.8
      },
      "past_races": [
        {
          "date": "0420",
          "place": "東京",
          "finish_position": "3",
          "indices": {
            "L4F": 82.1,
            "T2F": 70.5
          }
        }
      ]
    }
  ]
}
```

9. レースカード画面の仕様

9.1 表示項目

出走馬情報:

項目	説明
枠番	1〜8の枠番号（色分け表示）
馬番	出走番号
馬名	馬の名前
騎手	騎乗騎手名
斤量	負担重量（kg）
競うスコア	独自計算スコア（0〜100）

指数表示:

指数	表示位置	備考
L4F	過去走各行	後半4F指数
T2F	過去走各行	前半2F指数
ポテンシャル	過去走各行	潜在能力
レボウマ	過去走各行	激走可能性
巻き返し	過去走各行	復活期待度
クッション	過去走各行	馬場硬度

9.2 表示ロジックの注意点

0値の表示:

指数値が0の場合も正しく表示するため、以下のチェックを使用しています：

```
// 正しい実装
if (indices?.L4F != null) {
  // 表示処理
}

// 誤った実装（0が表示されない）
if (indices?.L4F) {
  // 0の場合にスキップされてしまう
}
```

キー名の統一:

APIレスポンスのキー名は英語で統一されています：

日本語名	英語キー
ポテンシャル	potential
レボウマ	revouma
巻き返し	makikaeshi
クッション	cushion

---

10. トラブルシューティング

10.1 指数が表示されない

原因1: race_idの不一致

```
# データベースのrace_idを確認
sqlite3 races.db "SELECT race_id FROM indices LIMIT 5;"

# 期待される形式: 202505040501001（18桁）
```

原因2: CSVファイルの形式エラー

- ヘッダー行が含まれている → 削除する

- 「作成用」ファイルが含まれている → 除外される（自動）
- 文字コードがUTF-8でない → UTF-8に変換

原因3: APIエンドポイントのキー名不一致

APIレスポンスのキー名が日本語になっていないか確認：

```
// 正しい
{ potential: 88.0, revouma: 6.5 }

// 誤り
{ ポテンシャル: 88.0, レボウマ: 6.5 }
```

10.2 CSVアップロードが失敗する

原因1: Node.jsがインストールされていない

```
# 確認
node --version

# インストール
# https://nodejs.org/ からダウンロード
```

原因2: サーバーが起動していない

```
# サーバー起動
cd /path/to/racescore-web
npm run dev
```

原因3: フォルダパスが異なる

`tools/upload-indices.ts` の `INDEX_FOLDERS` 定数を確認：

```
const INDEX_FOLDERS = [
  { name: 'L4F', path: 'C:\\競馬データ\\L4F\\2025' },
  // ...
];
```

10.3 競うスコアが0になる

原因1: 過去走データがない

```
sqlite3 races.db "SELECT COUNT(*) FROM umadata WHERE horse_name = '馬名';"
```

原因2: 巻き返し指数がない

過去走データに `index_value` カラムが設定されているか確認。

---

11. 運用チェックリスト

11.1 日次運用

- [ ] 当日の出走表データ（wakujun）をインポート
- [ ] 指数CSVファイルを所定フォルダに配置
- [ ] `sync-indices.bat` を実行して指数をアップロード
- [ ] レースカード画面で指数表示を確認

11.2 週次運用

- [ ] 過去走データ（umadata）の更新
- [ ] データベースのバックアップ（races.db）
- [ ] ログファイルの確認・削除

11.3 月次運用

- [ ] 不要な古いデータの削除
- [ ] データベースの最適化（VACUUM）

```
sqlite3 races.db "VACUUM;"
```

11.4 年次運用

- [ ] 年度フォルダの作成（例: 2026）
- [ ] `tools/upload-indices.ts` のフォルダパス更新
- [ ] 前年度データのアーカイブ

付録A: 主要ファイル一覧

ファイル	役割
`tools/upload-indices.ts`	指数CSVマージ＆アップロードスクリプト
`sync-indices.bat`	Windows用実行バッチファイル
`pages/api/upload-indices.ts`	指数アップロードAPIエンドポイント
`pages/api/race-card-with-score.ts`	レースカード＆スコアAPI
`utils/getClusterData.ts`	クラスタデータ・スコア計算ロジック
`lib/db-new.ts`	データベース接続・テーブル定義
`races.db`	SQLiteデータベースファイル

付録B: 技術スタック

カテゴリ	技術
フレームワーク	Next.js 14 (App Router + Pages Router)
言語	TypeScript
データベース	SQLite (better-sqlite3)
ORM	Drizzle ORM
スタイリング	Tailwind CSS
CSVパース	PapaParse

付録C: 変更履歴

バージョン	日付	変更内容
1.0	2025/05/04	初版リリース
1.1	2025/05/10	指数アップロード機能追加
1.2	2025/12/20	キー名統一、0値表示修正、マニュアル作成

本マニュアルに関するお問い合わせ:

本マニュアルの内容について不明点がある場合は、開発担当者にお問い合わせください。