ゲームデータ管理 DB システム仕様

– データベースによる安全で効果的なゲームデータ管理 –

2014 年 1 月 20 日 初版

板垣 衛

■ 改訂履歴

版	リリース	担当	改訂内容
初版	2014 年 1 月 20 日	板垣 衛	（初版）

■ 目次

■ 概略

ゲームデータをデータベースで管理するためのシステムを確立する。

なお、本書における「ゲームデータ」の定義については、「ゲームデータ仕様」を参照。

■ 目的

本システムは、ゲームデータ管理をより効率的で安全なものにすることを目的とする。

■ 従来のデータ管理手法の良い点と問題点

要件定義に先立ち、従来のデータ管理手法を再確認する。

● Excel 管理の良い点

- リッチな入力シートにより、着色やコメントなどを生かした分かり易いデータ編集が可能。
- 他者に影響を与えず、ローカルでデータを編集してテストすることが簡単。
- データのバックアップと、以前の状態への復元が簡単。

● データベース管理の良い点

- 複数名による同時編集作業が行いやすい。

● Excel 管理の問題点

- ファイル単位での編集ロックにより、複数名でのデータ編集がしにくい。
- 編集ロックのルールが守られなかった場合、データの巻き戻りが起こる。
- プログラムの都合に合わせたファイル分割が分かりにくい。
- ファイル分割の必要性により、同じテキストを複数のシートに書かなければならないことがある。
- テキストデータが分散し、校正の手間がかかることがある。
- テキストデータが分散し、多数の重複テキストで翻訳コストが無駄に高くなることがある。

- 構造変更時に（データ項目の追加など)、多数のファイルに対応する手間がかかることがある。

● データベース管理の問題点

- データが即時反映されてしまい、ローカルでのテストができない。
- 構造変更（データ項目の追加など）に非常に手間がかかる上、作業の手を止めてしまう。
- オフラインでデータ編集ができない。
- データの「バックアップ」と「以前の状態に戻す」操作が非常に行いにくい。
- 入力インターフェースが貧相でまともなコメント入力もできない。
- システムが大がかりになる上、メンテナンスに手間がかかる。
- システムに問題が生じると、全ての作業の手が止まるおそれがある。

■ 要件定義

▼ 基本要件

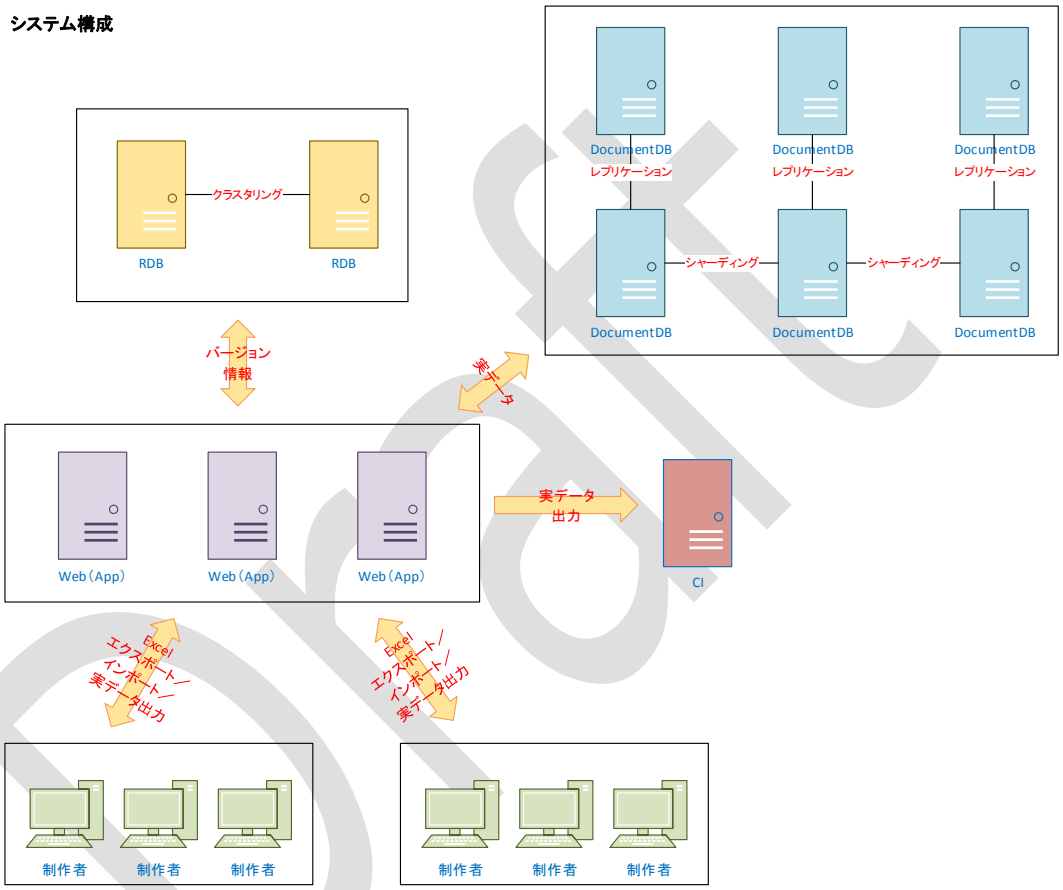
以上を踏まえ、システムの基本要件は下記の通り。

・ ゲームデータをデータベースで管理する。
・ データの入力は Excel のインポート／エクスポートで行う。
・ Excel シート上の装飾は、可能な範囲で維持する。
・ ランタイム時のファイル区分を気にせずにデータを入力できるものとする。
・ ピンポイントなデータのロックを可能とする。
・ バージョン管理の仕組みを実装し、ローカルでのテストや以前の状態へのロールバックに対応する。
・ 構造変更に柔軟に対応できるものとする。
・ 障害性に優れ、かつ、スケーラブルなシステムとする。
・ 複数タイトルのデータの一元管理をサポートする。
・ 優れた検索性を実装し、過去タイトルのテキスト照会も可能とする。これにより、重複テキストの検出やシリーズタイトルのテキスト照会を行い、翻訳コストの引き下げにも寄与するものとする。
・ Web ベースのツールで Excel のインポート／エクスポートを行うものとし、VPN を通じて外部のデベロッパや翻訳者も直接扱えるものとする。
・ ユーザーごとに扱えるタイトルが制限されるなどのセキュリティも備える。

▼ 要求仕様／要件定義

以下、本書が扱うシステムの要件を定義する。なお、要件として不確定の要求仕様も併記する。

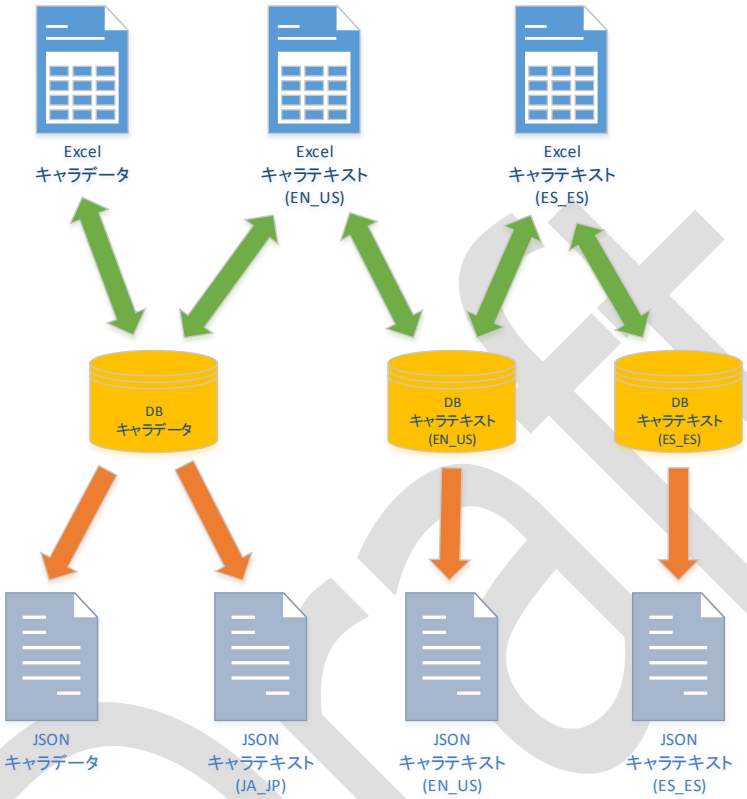
・ RDB サーバー＋ドキュメント指向 DB サーバー＋Web（アプリケーション）サーバーで構成する。



- RDB では、バージョンとデータの階層・位置情報の管理のみを行い、実データは記録しない。タイトル依存のテーブルはないため、稼働後の構造変更は基本的に発生しない。PostgreSQL の利用を想定。（ユーザー管理、プロジェクト管理、権限管理も RDB）
- ドキュメント指向 DB に実データを記録する。MongoDB の利用を想定。純粋な JSON 形式のデータ（BSON 形式データ）が格納される。
- Web サーバーに実装する。Excel のエクスポート／インポート処理、および、実データ（JSON）のエクスポートを行う。ASP.Net の利用を想定。
- RDB とドキュメント DB はクラスタリング／レプリケーションによる耐障害性の確保を行う。
- ドキュメント DB と Web サーバーはスケールアウト可能とする。Web サーバーはラウンドロビン DNS と組み合わせて使用する。

- CI サーバーによる、全データの出力とオーサリングの自動実行を行う。Jenkins の利用を想定。

・ DB 上のデータ、Excel 上のデータ、出力される実データは、それぞれ異なる構造となることを可能とする。



DB キャラデータ：

```
[
      { "ID": "c0010", "Name": "山田", "HP": 100, "ATK": 12, "DEF": 30 },
      { "ID": "c0020", "Name": "田中", "HP": 110, "ATK": 13, "DEF": 25 },
      { "ID": "c0030", "Name": "佐藤", "HP": 120, "ATK": 9, "DEF": 40 },
]
```

DB キャラテキスト(EN_US) ：

```
[
      { "ID": "c0010", "Name": "Yamada" },
      { "ID": "c0020", "Name": "Tanaka" },
      { "ID": "c0030", "Name": "Ssto" },
]
```

DB キャラテキスト(ES_ES) ：

```
[
      { "ID": "c0010", "Name": "Yàmàdà" },
      { "ID": "c0020", "Name": "Tànàkà" },
      { "ID": "c0030", "Name": "Sàto" },
]
```

Excelキャラデータ

ID	名前	パラメータ		
		HP	ATK	DEF
c0010	山田	100	12	30
c0020	田中	110	13	25
c0030	佐藤	120	9	40

Excelキャラテキスト(EN_US)

ID	Name	
	JA_JP	EN_US
c0010	山田	Yamada
c0020	田中	Tanaka
c0030	佐藤	Sato

Excelキャラテキスト(ES_ES)

ID	Name	
	EN_US	ES_ES
c0010	Yamada	Yàmàdà
c0020	Tanaka	Tànàkà
c0030	Sato	Sàto

JSON キャラデータ：（実データ）

```
[
      { "ID": "c0010", "HP": 100, "ATK": 12, "DEF": 30 },
      { "ID": "c0020", "HP": 110, "ATK": 13, "DEF": 25 },
      { "ID": "c0030", "HP": 120, "ATK": 9, "DEF": 40 },
]
```

JSON キャラテキスト(JA_JP) ：（実データ）

```
[
      { "ID": "c0010", "Name": "山田" },
      { "ID": "c0020", "Name": "田中" },
      { "ID": "c0030", "Name": "佐藤" },
]
```

JSON キャラテキスト(EN_US) ：（実データ）

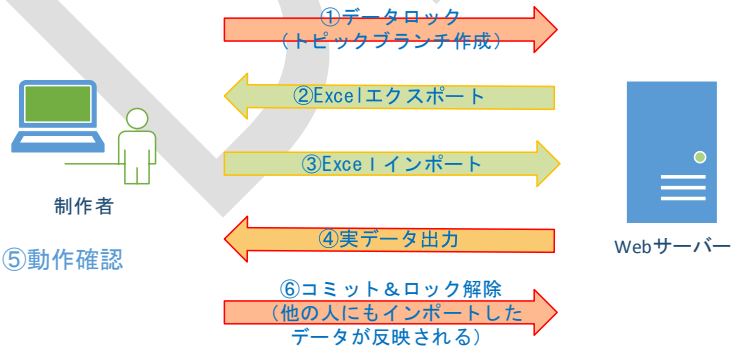
```
[
      { "ID": "c0010", "Name": "Yamada" },
      { "ID": "c0020", "Name": "Tanaka" },
      { "ID": "c0030", "Name": "Ssto" },
]
```

JSON キャラテキスト(ES_ES)：（実データ）

```
[
      { "ID": "c0010", "Name": "Yàmàdà" },
      { "ID": "c0020", "Name": "Tànàkà" },
      { "ID": "c0030", "Name": "Sàto" },
]
```

・ Web インターフェースでは、下記の操作をサポートする。

制作者の大まかなワークフロー



- ユーザーログイン
- 対象プロジェクトの選択
- ローカル作業の開始（データのロックとトピックブランチの作成）
  - トピックブランチとは、ローカル作業用の一時的なブランチのこと。
  - トピックブランチを本流に反映（コミット）させない限りは、インポートしたデータが他者に

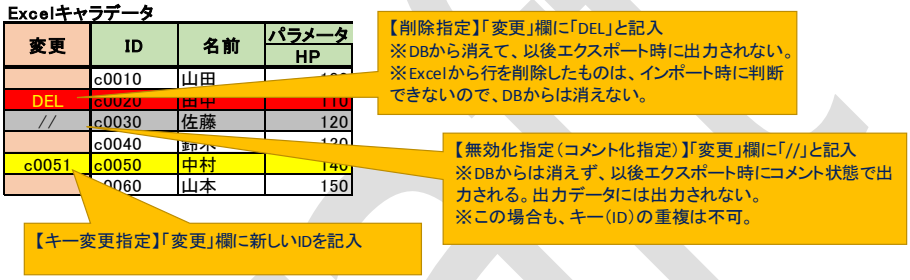
は反映されない。

- 対象データがロックされるため、他者が同じデータを編集できない。
- データのロックは、広範囲にも行単位（ピンポイント）にも行うことが可能。
- トピックブランチ作業中に、後からロック対象データを追加することが可能。
- トピックブランチ作業中に、一部のデータを過去バージョンに戻すことが可能。

➢ Excel のエクスポート（対象データを指定）

➢ Excel のインポート

- インポート時には、Excel 上に記録されている情報から、データの種別を判別して処理する。
- インポートされたデータを照合して、変更のあったデータが画面にリストアップされる。
- データを削除したい場合や、キー（ID）を変更したい場合は、Excel の専用覧に記入してインポートする。



➢ 実データ（JSON テキスト）の出力

- バイナリデータへの変換はローカルで行う。

➢ ローカル作業の完了

- トピックブランチの本流への反映（コミット）もしくは破棄
- ロック解除
- コメントの記録

➢ 変更履歴の確認

・ Excel の入力補助機能に対応。

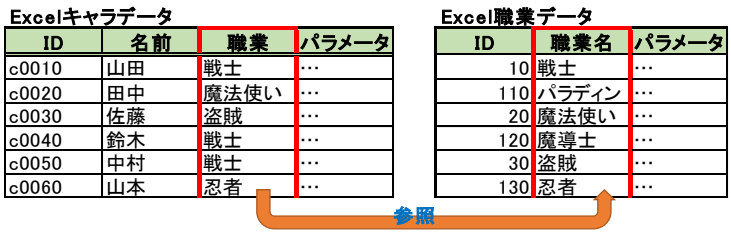
➢ 例えば、Excel 上では「職業」として「戦士」「魔法使い」などと分かり易い名称で入力するが、実データとして出力される際は、「1」「2」などの値に変換される。

➢ 参照するデータも DB 上に記録されている。

➢ Excel の対象項目でダブルクリックもしくは右クリックすると、選択候補がリストアップされる。

- この選択用のリストは、Excel エクスポート時に別のシートにいっしょに出力される。制作者の PC から DB に直接アクセスするようなことはない。そのため、オフライン作業も可。

➢ 変換できないデータが指定されたら、Excel インポート時にエラー報告し、インポートに失敗する。



DB キャラデータ：

```
[
    { "ID": "c0010", "Name": "山田", "Class": "戦士" },
    { "ID": "c0020", "Name": "田中", "Class": "魔法使い" },
    { "ID": "c0030", "Name": "佐藤", "Class": "盗賊" },
    { "ID": "c0030", "Name": "鈴木", "Class": "戦士" },
    { "ID": "c0030", "Name": "中村", "Class": "戦士" },
    { "ID": "c0030", "Name": "忍者", "Class": "忍者" },
```

DB には入力値のまま記録される。

JSON キャラデータ：（実データ）

```
[
    { "ID": "c0010", "Name": "山田", "Class": 10 },
    { "ID": "c0020", "Name": "田中", "Class": 20 },
    { "ID": "c0030", "Name": "佐藤", "Class": 30 },
    { "ID": "c0030", "Name": "鈴木", "Class": 10 },
    { "ID": "c0030", "Name": "中村", "Class": 10 },
    { "ID": "c0030", "Name": "忍者", "Class": 130 },
```

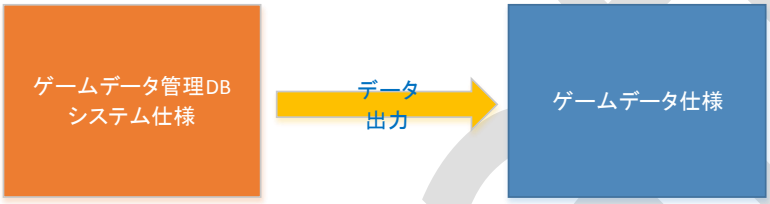
出力データでは ID に変換される。

- DB 上の「データ構造定義」、「Excel の構造定義」、「DB と Excel の変換設定」、「Excel の入力補助設定」、「DB から出力データの変換設定」は、JSON 形式の設定データとして、DB に記録して扱う。
- テキストデータに関しては、データ定義の違いやタイトルを超えて、全テキストデータの検索や重複データのピックアップなどを行う機能をサポートする。
  - 例えば、キャラの名前が変更になった時に、古いキャラ名が使われている箇所を全てピックアップしたり、翻訳時に過去タイトルの翻訳に合わせたり、固有名詞の翻訳に一貫性があるかをチェックしたりする場合に活用する。
- DB 上のデータは、下記のような階層的な区分で管理する。
  - **タイトルグループ**
    - 例：「〇〇の冒険」シリーズ
  - **タイトル**
    - 例：「〇〇の冒険」「〇〇の冒険２」
  - **バージョン**
    - 例：「オリジナル版」「完全版」「海外版」
  - **【任意】データグループ**
    - 例：「キャラ系」「マップ系」「戦闘系」

- **データ定義**
  - 例：「キャラパラメータ」「戦闘パラメータ」
  - このレベルでデータ構造が定義される
- **【任意】データカテゴリ**
  - ※「ステージ別」「章別」「マップ別／プランナー用」「マップ別／マップ班用」

・ユーザーごとに、タイトルへのアクセス権限が設定される。
- タイトルグループ全体にアクセス権限を適用して、過去タイトルのデータの参照を許可することも可能。

▼ 仕様の依存関係



DB からの出力データの仕様は、「ゲームデータ仕様」に準拠する。

■ データ仕様

（未定）

【構想】

バージョン管理とデータ本体の管理手法は、Git の管理構造を模倣する。

ファイル内容から算出した 160 ビットの SHA-1 値をキーにし、レコード（Excel の 1 行に相当）ごとにデータを記録する。

前述のデータの階層区分をディレクトリ、レコードをファイルに見立てて管理する。

Git と同じく、「ツリーオブジェクト」（フォルダとファイルのリストを管理）、「ブロブオブジェクト」（レコード内容を管理）、「コミットオブジェクト」（コミット時のリビジョンやコメントを管理）で管理し、これらの情報は RDB に格納する。

「ブロブオブジェクト」の実体が、ドキュメント DB に記録される。さらに、データ変換の定義情報も「ブロブオブジェクト」の一種として管理する。

■ 処理仕様

（未定）

■■

■ 索引

索引項目が見つかりません。

ゲームデータ管理 DB システム仕様

以　上