kintoneを使った設計・構築まとめ

kintoneアプリデザインスペシャリストのベースとなるkintoneSIGNPOSTのサイト情報を整理したもの

修正の履歴

Version	更新日	内容
1.00	2024.07.01	新規作成
2.00	2024.07.16	「4-2.データ連携方法の整理」「4-3. 変化に強いkintoneのシステムを考える」を追加

1. kintoneSIGNPOST

1-1. 構築概要

- kintoneの基本機能（プラグイン含む）で実現出来る方法を検討
  カスタマイズは可能な限り減らす
- 運用の中で各種権限設定や方針が見えてくるので、最初は緩いガバナンスでアイデアを集める方を優先
- 閲覧権限は「見せてはいけない情報」を閲覧不可とし、それ以外はオープンにすることで、アクセス権のメンテナンスをカンタンにしておく
  ※　不必要なアクセス権はkintoneのパフォーマンスにも悪影響を及ぼす
- 初回設計で厳格にすると、保守性が悪くなるので、継続して見直しをする前提て考える
- 運用ミスでのデータ損失を確実に防ぐ為、損失が許されない業務データを洗い出し、重要性（影響度・更新頻度）に応じて業務データの管理体制を整備
- 外部からデータを持って来る場合は、不要なデータをできるだけ削る
- 業務アプリのリリースは小さい単位で提供する

1-2. 設計

- 社内外メンバーが協業する場合、同一ドメインを前提とした運用ルール作りを検討
  パターン別の設計方法はパターン実装ガイド参照

- 情報は「ストック情報」と「フロー情報」に分けて検討
  「ストック情報」：特定の型によって蓄積され、分析や再利用の対象になるもの
  「フロー情報」　：特定の型を持たず、一時的な共有を目的とするもの
  業務システムの基本はストック情報だが、アプリ間でのコメントや質問等の一時的な情報はフロー情報で管理し、全てをストックにする必要は無い

- ワークフロー等プロセスは可能な限りシンプルに
  設計の詳細はパターン実装ガイド参照

- 同一のデータを複数の役割で利用する場合、1つのアプリで複数の役割を想定するより役割毎にアプリを構築するほうが使いやすい

- 複数のアプリで共通となるI/F（機能）であれば、「親子アプリ」として設計する
  1:Nになるような構成（機器1件に対して、貸出実績が複数件）であれば、機器情報が親アプリ、貸出情報が子アプリとなる
  親子のアプリを繋げる方法として「ルックアップ」「関連レコード一覧」「アプリアクション」がある

- アクセス権の設定は下記順番で行うことで軽量化を図る

  1. 所属スペースの設定 or アプリグループの設定
  2. アプリのアクセス権
  3. レコードのアクセス権
  4. フィールドのアクセス権（グループフィールドによる一括設定）
  5. フィールドのアクセス権（個別フィールド）

- アクセス権の設定対象は下記順番で行う

  1. グループ or 組織
  2. ユーザー

- kintoneのアップデートを考慮して、UI/UXの大幅なカスタマイズは可能な限り行わない（現場の「使いづらい」は慣れや、簡単な見直しで解決することが多い）

- マスタアプリは全体で共通利用するアプリとして早めに作成

- kintoneの性能を考慮して「**データ量×複雑度×アクセス数**」を考える
  単体で考慮するのではなく複合的に考える必要あり

  1. データ量
     - レコード数は100万件が目安
       ただし、ソート条件やアクセス権の複雑さで数万レコードでも性能に影響が出ることがある
       ※　ストック情報（保管対象）から他にエクスポートして削除できるものが無いか検討
       ※　実際使っていないフィールドが存在しないか検討
       ※　年度やエリア情報を別画面に分けられる場合はアプリを分割することを検討（単純に件数で分けられるならそれもあり）
     - 1アプリのフィールド数は100を目安とする 1アプリの最大フィールド数は500だが、100を越えると遅延が発生。特に「テーブル、関連レコード一覧が多い」「テーブルのフィールド数や行数が多い」には注意が必要
       ※　フィールド数が多い場合、親子アプリに分けてリレーションで継げるか検討

※　レコード一覧表示のフィールド数は削減できるか検討
※　データ量が多いリッチエディターは一覧表示させないことを検討

2. 複雑度
   - 絞り込み・ソートの条件はできるだけ簡単にすること。「or」「not」「リッチエディターの検索」「ドロップダウンやラジオバタンのソート」は遅延の要因
   - アクセス権の設定条件が複雑な場合
     ※　レコード一覧のデフォルト表示は条件をシンプルにしておき、複雑な条件の一覧は切り替えて表示する運用を考える
3. アクセス数
   - ユーザーアクセスの集中（始業終業時等）は遅延の要因
   - kintone REST APIの同時接続が集中する場合は遅延の要因。kintone REST APIの同時接続数の上限は1ドメイン100まで
     ※　「お知らせ」のアプリは必要最低限にする（アクセス時にアプリへのリクエストが発生するので、数はできるだけ少なくしておく）
     ※　アクセス時のAPIリクエストは控えて、イベントよる取得等仕組みを考える
     ※　一時データを再利用可能であればアプリに保存して再利用する
     ※　同じユーザーが何度もアクセスする場合は、データをセッションのストレージに保存する

参考サイト : /home/smds/smds_optiserve/poc/docs

1-3. 開発ルール

- アプリ作成のルールを社内で決定する
  - アプリ名の命名規則
  - アプリの責任者と保守担当の設定、明記
  - 現場メンバーのアプリ作成権限の制御
  - アプリ作成・公開時の運用ルール

2. kintone技術まとめ

2-1. アクセス権の事前評価

①　経緯

- kintoneの速度遅延問題としてアクセス権に伴う問題があり改善が行われている。
  - Step1. リアルタイムでアクセス権の評価をするとレコード取得時間の遅延問題あり
  - Step2.「アクセス権の事前評価」処理で高速なレコード参照を実現、ただしレコード件数やアクセス権の設定件数によってはアプリの更新時の遅延が発生
  - Step3. アクセス権の設定がアプリ更新時に順次反映されるように変更

② アクセス権の事前評価とは

- レコードを保存（追加や編集）時、"アクセス権の設定されたレコード / フィールド" と "アクセス権の設定されたユーザー / 組織 / グループ" の組み合わせで決まる**アクセス権を事前に計算**して、データベースに保存。
- これにより "アクセス先のレコード / フィールド" と "ログインユーザーの情報" を元に事前評価結果を調べれば、即座にアクセス権の有無が判明

③ デメリット

- 「アプリの更新」に時間がかかる
  レコード数やアクセス権の設定件数が多いトアプリの更新に時間がかかる
  <事前評価結果が再計算されるタイミング>

1. アプリの設定画面でレコード / フィールドのアクセス権を変更し、「アプリの更新」を実行した場合
2. kintone REST APIでテスト環境のレコード / フィールドのアクセス権を変更し、テスト環境の設定を運用環境に反映した場合
3. kintone REST APIで運用環境のレコード / フィールドのアクセス権を変更した場合

- アクセス権の事前評価の処理中にデータベースロックエラーとなる処理がある
  アクセス権の事前評価結果を再計算している間は、データベースの書込み競合を防ぐため「レコードの追加・更新」「アプリの設定変更」が出来なくなる。
  この時画面には「データベースロックエラー」が表示される

④ 参考サイト

アクセス権の事前評価とアプリの更新時間について

2-2. データ連携方法の整理

特定の情報をキーに他のデータと連携する方法は複数存在する。
どれを利用するかメリットとデメリットを整理。

①　テーブル

データが基本的に1：Nの関係になっている場合はデータを利用。
行数を増やせる運用が可能となることや、集計関数等も簡単に利用可能なのがメリット。
デメリットとしては次のとおり。

- 一度テーブルに設定すると、元に戻せない
- テーブル内のフィールドの値は他のデータやアプリから参照出来ない

② ルックアップ

特定のキーを利用してマスタの情報をセットすることで入力効率を高める場合に利用。
DBアプリを構築する場合、マスタ情報を取得したトランザクションの情報はトランザクションに値としてセットされるので、マスタの値が変更になっても変わることが無く、kintoneも基本的には同じ。
ただしアプリの「取得」ボタンがある為、最新に書き換えることも可能なので、情報管理としては方針をきちんと決めておく必要あり。
また基本的な機能としては「取得」を必ずクリックしてデータ取得を行う必要があり、値が入力されたら自動で値を取得してセットするにはJavaScriptで作成する必要あり。
※ プラグインも発売されている

他のデメリットは次のとおり。

- 指定出来るフィールドは「文字列1行」「数値」「計算」「ルックアップ」「リンク」「レコード番号」のみ
- チェックボックスやラジオボタン、リスト等は利用出来ない
- テーブルに設定したフィールドは指定できない

③ 関連レコード一覧

特定のキーを利用して、別のアプリの情報を参照することが可能。
取得する情報はフィルターによる選択や取得件数の指定も可能。

デメリットは次のとおり。

- 一覧画面には表示されない
- 「集計」「自動計算」「アプリ内検索」は非対応
- 表示出来るフィールドは「文字列1行」「数値」「計算」「ルックアップ」「リンク」「レコード番号」のみ
- ファイルに書き出す場合、関連レコード一覧の項目は出力対象外
- テーブルに設定したフィールドは指定できない
- 表示されるフィールドの横幅調整はできない（自動のみ）

2-3. 変化に強いkintoneのシステムを考える

kintoneは「速い・簡単」にアプリを作成することが出来るが、変化に強いアプリとなるとコツがある。
変化に強いアプリの定義を次の2つとする。

- 拡張・連携したいと思ったときに容易に可能
- データのバックアップで一括更新（リストア）が可能

この要点を満たす為に初回開発時から対応しておく事

- **レコード作成時にユニークとなる「レコード番号」以外に連番となるuidを用意する**:
  - kintoneの場合、PrimaryKeyをレコード番号に持たせ、複数項目をPrimaryKeyにするような考えは存在しない。サービス任せでなく独自のPrimaryKeyとなる値を準備しておくと再利用しやすい
- **連携に必要なルックアップキーを用意**:
  - 複数項目を利用したルックアップが出来ない為、必ずルックアップのキーとなるフィールドを用意しておく必要がある、複数項目がキーになる場合は、それらを連携した項目をあらかじめ用意しておくことで利用しやすくなる
- **マスタになりそうな場合、名称がユニークだとしても必ず関連するコードを用意**:
  - kintoneの考えよりRDBの考えで設計するイメージ
- **作業レコードでもユニークとなるような項目「レコードタイトル」を準備しておく**:
  - IDではなくキーとなる項目の組み合わせになるようにしてキーとして利用できるようにしておく

3. 参考サイト

3-1. サンプルソース

この節の説明の元となったサイトにてこの考えを実現する為のjavascriptをgithubで公開している。 セットで、kintoneで便利な機能を集めたjavascriptも公開している。

サイト情報	URL
カスタマイズアプリテンプレート	https://github.com/AISIC-DL/downloads/blob/master/template/カスタマイズアプリテンプレート.zip
kintone_tools	https://github.com/j-kume/kintone_tools

3-2. 参考サイト

サイト情報	URL
ラーニングメイン	https://cybozu.co.jp/kintone-certification/training/?tab=ds
kintoneSIGNPOST	https://kintone.cybozu.co.jp/kintone-signpost/?_gl=11*pzeh0k*_gcl_au*MjY4NDU3NTA2LjE3MTk3OTI0NTY.*_ga*MTg0OTYwMzU3MC4xNzE5NzkyNDU2*_ga_T5K95WXL54*MTcxOTc5MjQ1Ni4xLjEu
パターン実装ガイド	https://kintone.cybozu.co.jp/kintone-signpost/guide/

以上