RDBアンチパターンと戦う

- 削除フラグ 完全攻略ガイド -

PHPカンファレンス関西

新連載

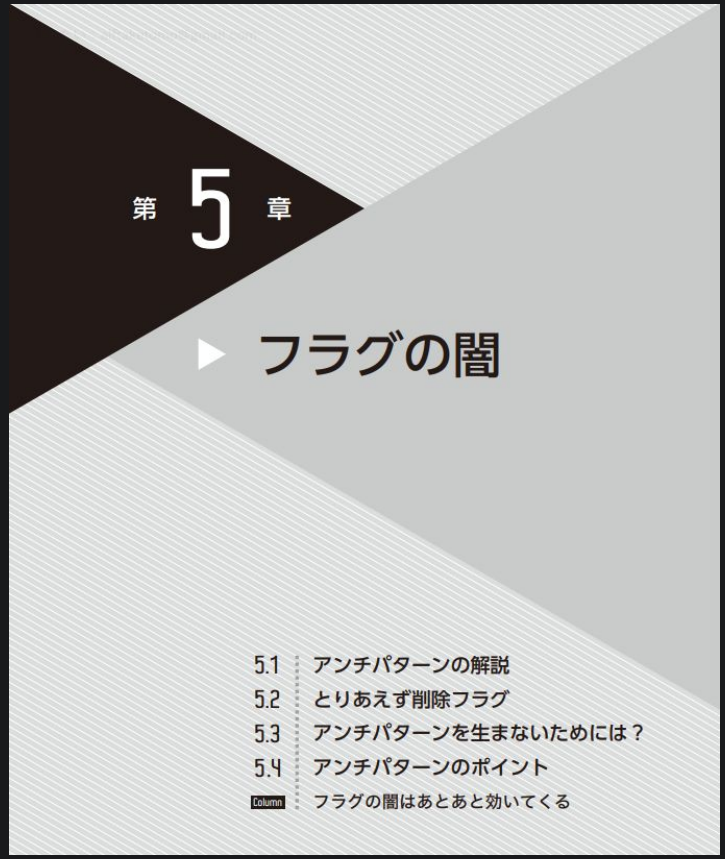
実践データベースリファクタリング

アンチパターンに立ち向かう　第1回

削除フラグに立ち向かう

曽根 壮大
SONE Taketomo
(同)Have Fun Tech/ (株)Linkage
GitHub soudai
Twitter @soudai1025


Software Design plus
失敗から学ぶ
RDBの
正しい歩き方
曽根 壮大
RDB The Right Way
データベースの迷宮
失われた事実
やり過ぎたJOIN
効かないINDEX
フラグの闇
ソートの依存
隠された状態
JSONの甘い罠
強過ぎる制約
転んだ後のバックアップ
見られないエラーログ
監視されないデータベース
知らないロック
ロックの功罪
簡単過ぎる不整合
キャッシュ中毒
複雑なクエリ
ノーチェンジ・コンフィグ
塩漬けのバージョン
フレームワーク依存症
MySQL・PostgreSQLの
設計と運用を見直す
技術評論社


第 5 章
フラグの闇
5.1 アンチパターンの解説
5.2 とりあえず削除フラグ
5.3 アンチパターンを生まないためには？
5.4 アンチパターンのポイント
Column フラグの闇はあとあと効いてくる

What is it?

この話は

上の2冊でも紹介しています

What is it?

データベースの寿命は

アプリケーションより長い

What is it?

削除フラグは

はアンチパターン

What is it?

それを知っていても

どう立ち向かえばいいかわからない……

What is it?

そんな人のために

削除フラグとの戦い方をお伝えします

あじぇんだ

1. 自己紹介
2. アンチパターン：とりあえず削除フラグ
3. あるべき姿
4. 移行先のテーブルとダブルライト
5. 子テーブルへ分割
6. まとめ

あじぇんだ

1. 自己紹介
2. アンチパターン：とりあえず削除フラグ
3. あるべき姿
4. 移行先のテーブルとダブルライト
5. 子テーブルへ分割
6. まとめ

自己紹介

曽根(そね) 壮大(たけとも)（39歳）

Have Fun Tech LLC 代表社員
株式会社リンケージ CTO

- 日本PostgreSQLユーザ会 勉強会分科会 担当
- 3人の子供がいます（長女、次女、長男）
- 技術的にはWeb/LL言語/RDBMSが好きです
- コミュニティが好き

luccafort
@luccafort

同僚になった人の自己紹介を読んでいたら唐突に @soudai1025 の著書「失敗から学ぶRDBの正しい歩き方」のおかげで未経験文系だったけどエンジニアとして独り立ちできました！ってあったのでほほぅ、そーだいすげーじゃん？ってなった。

失敗から学ぶRDBの正しい歩き方 (Software Design plus)
失敗から学ぶRDBの正しい歩き方 (Software Design plus)
amazon.co.jp

突然の宣伝

予防医療のリンケージ

- リモートワークの不安を数値にするストレスチェック
- 女性の健康課題をサポートする
- リモートワーク、ちょっとした心配を相談できる安心をご提供

あじぇんだ

1. 自己紹介
2. アンチパターン：とりあえず削除フラグ
3. あるべき姿
4. 移行先のテーブルとダブルライト
5. 子テーブルへ分割
6. まとめ

とりあえず削除フラグ

削除フラグの引力

とりあえず削除フラグ

ことのはじまり

とりあえず削除フラグ

id	name	created_at	delete_flag
1	Taro	2023-03-10 12:00:00	false
2	Hanako	2023-03-11 14:00:00	false
3	Jiro	2023-03-12 10:30:00	true
4	Yoko	2023-03-13 16:45:00	false
5	Ken	2023-03-14 09:15:00	true

とりあえず削除フラグ

Aさん：あー、テーブルは論理削除を使っているね。delete_flagという削除フラグが付いている。

Bさん：O/Rマッパに前任の人が独自に実装した機能が追加してあって、この削除フラグはWHERE句に自動的に付与されるんだよね。

Aさん：なるほど。メジャーなO/Rマッパでも論理削除を使える拡張もあるし、これで問題はなさそうだね。

とりあえず削除フラグ

数年後……

とりあえず削除フラグ

Cさん：データがすごく大きくなったから、削除フラグがtrueのデータを消そうかな。

Aさん：ダメダメ！ 削除フラグがtrueのデータも参照しているから消せないよ。

Cさん：削除フラグなのに削除できないんですか……。

とりあえず削除フラグ

アンチパターンは

良かれと思ったことが後から問題になる

とりあえず削除フラグ

削除フラグの問題

とりあえず削除フラグ

- クエリが複雑になる
- パフォーマンスが低下する
- 制約が活用できず整合性を犠牲にする

クエリが複雑になる

削除フラグがある場合に有効なユーザーを取得したいときのSQLは、`SELECT * FROM user WHERE delete_flag = false`になります。

では、ユーザーに紐付く会社や組織にも削除フラグがあった場合はどうでしょうか。

所属している会社が削除されているのにユーザーが取得できることはバグの温床になり、たいへん危険です。そのため、次のSQLになります。

```
SELECT
 *
FROM
 user
 -- customerをJOINして確認
 INNER JOIN customer AS c
 ON user.id =c.user_id
 AND c.delete_flag = false
WHERE user.delete_flag = false;
```

とりあえず削除フラグ

INDEXの利用が難しいため

パフォーマンスが劣化する

とりあえず削除フラグ

制約が活用できない

とりあえず削除フラグ

id	name	created_at	delete_flag
1	Taro	2023-03-10 12:00:00	false ←削除されたユーザー
2	Taro	2023-03-11 14:00:00	true ←同名だがidが違うので別ユーザーの扱い
3	Jiro	2023-03-12 10:30:00	true
4	Yoko	2023-03-13 16:45:00	false
5	Ken	2023-03-14 09:15:00	true

とりあえず削除フラグ

これらの要素が

トラブルの温床になる

あじぇんだ

1. 自己紹介
2. アンチパターン：とりあえず削除フラグ
3. あるべき姿
4. 移行先のテーブルとダブルライト
5. 子テーブルへ分割
6. まとめ

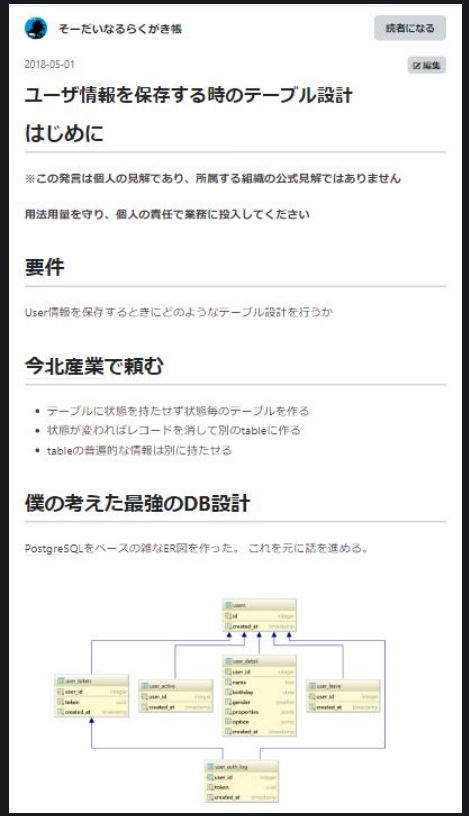
あるべき姿

削除フラグが無い姿を目指す

あるべき姿

削除フラグが無い姿を目指す
↓
1つのテーブルに1つの責務の状態にする

あるべき姿



そーだいなるらくがき帳

読者になる

2018-05-01

編集

ユーザ情報を保存する時のテーブル設計

はじめに

※この発言は個人の見解であり、所属する組織の公式見解ではありません

用法用量を守り、個人の責任で業務に投入してください

要件

User情報を保存するときにどのようなテーブル設計を行うか

今北産業で頼む

- テーブルに状態を持たせず状態毎のテーブルを作る
- 状態が変わればレコードを消して別のtableに作る
- tableの普遍的な情報は別に持たせる

僕の考えた最強のDB設計

PostgreSQLをベースの雑なER図を作った。これを元に話を進める。

https://soudai.hatenablog.com/entry/2018/05/01/204442

あじぇんだ

1. 自己紹介
2. アンチパターン：とりあえず削除フラグ
3. あるべき姿
4. 移行先のテーブルとダブルライト
5. 子テーブルへ分割
6. まとめ

あじぇんだ

1. 自己紹介
2. アンチパターン：とりあえず削除フラグ
3. あるべき姿
4. 移行先のテーブルとダブルライト
5. 子テーブルへ分割
6. まとめ

リファクタリング

WEB+DB PRESS Vol.134

実践データベースリファクタリング
を読んでくれ！

あじぇんだ

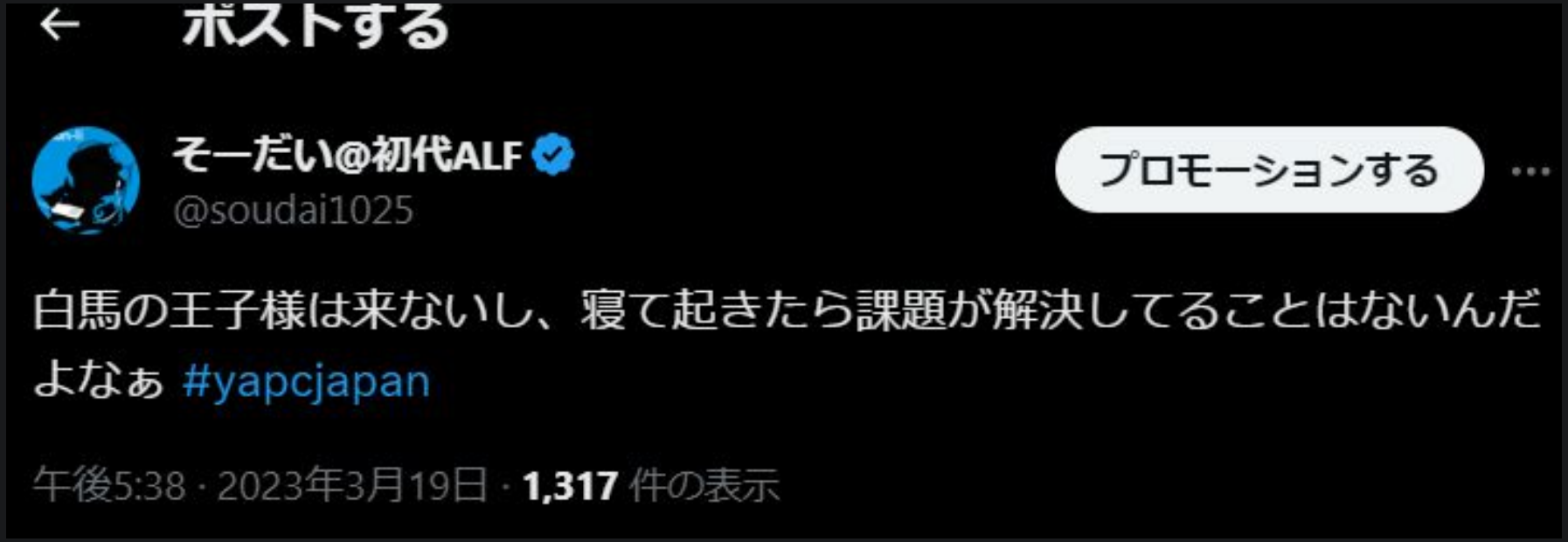
1. 自己紹介
2. アンチパターン：とりあえず削除フラグ
3. あるべき姿
4. 移行先のテーブルとダブルライト
5. 子テーブルへ分割
6. まとめ

まとめ

リファクタリングは

日常的に行う

まとめ


ポストする
そーだい@初代ALF
@soudai1025
プロモーションする
白馬の王子様は来ないし、寝て起きたら課題が解決してることはないんだよなぁ #yapcjapan
午後5:38 · 2023年3月19日 · 1,317 件の表示

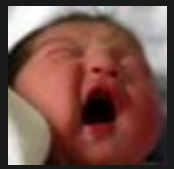
まとめ

失敗を恐れるのではなく

失敗できるようにする

人生を変えるために必要なこと

“手を動かした者だけが、世界を変える”



株式会社はてな id:onishi

まとめ

ご清聴ありがとうございました