RDBでCDB

山ノ内

自己紹介

名前	山ノ内　祥訓(よしのり)
年齢	30歳になったばかり （8月29日生まれなので・・・）
居所	熊本県
会社	熊本市内にあるKISという会社です
主なお仕事	電子カルテなどなど医療関連のシステム導入をメインでやっているSIの人 たまにインフラの構築とか開発とかをやってることもあり 最近他県への遠征多し

今日のお話

最近クラウドとかの話で出てくるGoogleのBigtableについて
興味があったのでいろいろ調べていたら

「もしかして普通のDBで近いものがつくれるじゃね？」

と何を間違ったか思ってしまった結果、
どんなものができたかを紹介します。

そもそもなんでこんなことをやってみようと思ったのか

医療関係のデータ(特に患者属性情報)を効率よく扱いたい・・・

★　時系列的に把握が必要なデータが多い。
　　⇒身長、体重、バイタル、薬歴、検査歴など。

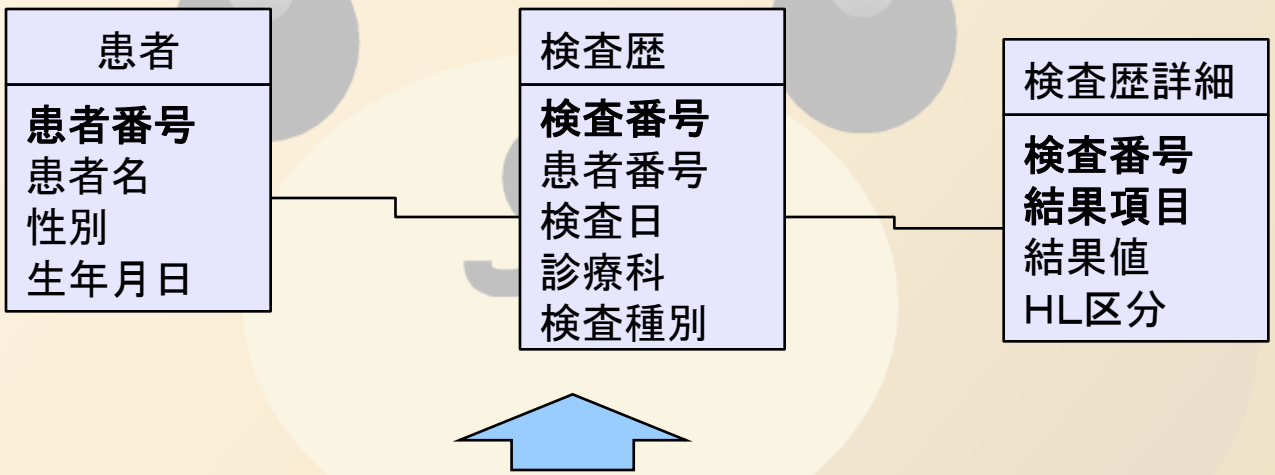
★　ほぼ全てのデータは履歴管理が必要。
　　⇒でも、そのために行データを複製する？

★　ほとんどの場合キー検索にしかならないデータ構造
　　⇒患者ID＋日付の検索で事足りることが多い。

★　外にデータを持つという部分についての問題。
　　⇒診療情報を外に保存するのはと～ってもハードルが高い。

まずはおさらいRDBMSについて

1. データ構造として行と列の2次元のテーブルをもちます。
2. 正規化により関連付けられた複数のテーブルへデータを分割して格納します。
3. データの管理および操作にはSQLを使用します。




```
SELECT TH.検査番号,TH.検査日,TH.診療科,TH.検査種別,TH.患者番号,
       P.患者名,TD.結果項目,TD.結果値,TD.ＨＬ区分
FROM 検査歴 TH ,検査歴詳細 TD,患者 P
WHERE TH.検査番号=TD.検査番号 AND TH.患者番号=P.患者番号
```

Bigtableについて

GoogleのWebサービスで使用している分散データベースのことです。
数百～数千台のサーバを並列に動かしてペタバイト(10^15)クラスの
データを扱っているそうです。
去年からこのDBはGoogle App Engine (GAE) として
一般ユーザも使用できるようになりました。

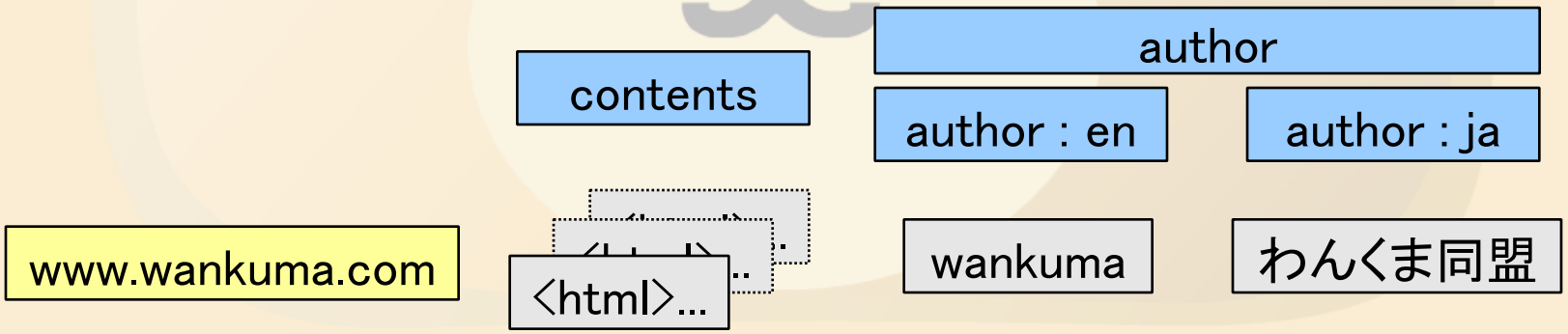
Bigtableのデータモデルや構造については、

http://labs.google.com/papers/bigtable.html

で論文を読むことができますので、これを手がかりに進めていきます。
(ちなみに英語なので・・・)

Bigtableの特徴1

1. 全体的な構造としてはRDBMSと同じ行と列で構成されています。ただし、列は行ごとに変えることができます。
2. データ(Cellというらしい)はバイト配列で格納されます。
3. 全てのデータはタイムスタンプを持っているためバージョン管理されます。KVSとしては行キー(RowKey)＋列キー(ColumnKey)＋タイムスタンプ(Timestamp)をキーとして扱います。
4. ひとつのカラムに対して複数の子要素を指定できます(ColumnFamily)。
5. SQLはサポートされていません。

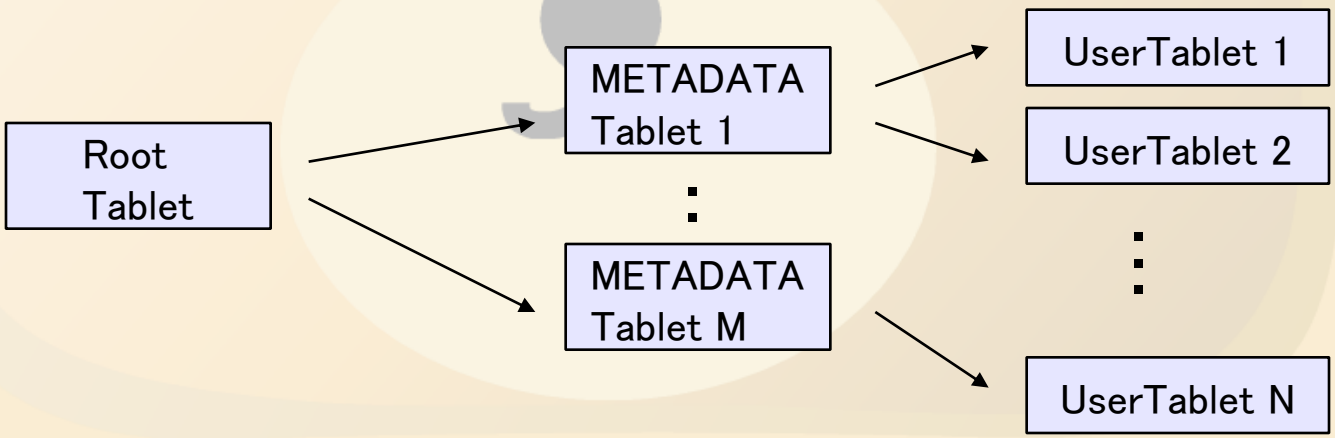


Bigtableの特徴2

行データのかたまり(Tablet)ごとに分けてから
各サーバに格納されます。かたまりの大きさは100～200MB。

データが入ったTabletを束ねるのがMETADATATabletで、全ての
METADATATabletの管理情報を持っているのがRootTablet。

クライアントは積極的にMETADATAをキャッシュすることが
求められているみたい。



今回の環境について

今回使った環境は・・・

データベース・・・PostgreSQL
開発環境・・・VisualStudio2005 C#(.NET2.0)

なんでポスグレ

日本医師会総合政策研究機構ORCAプロジェクト

http://www.orca.med.or.jp/

ここでオープンソースとして提供されている医療事務システムである
ORCAが使用しているデータベースがPostgreSQL。
なんでこれと同居できるようにしています。

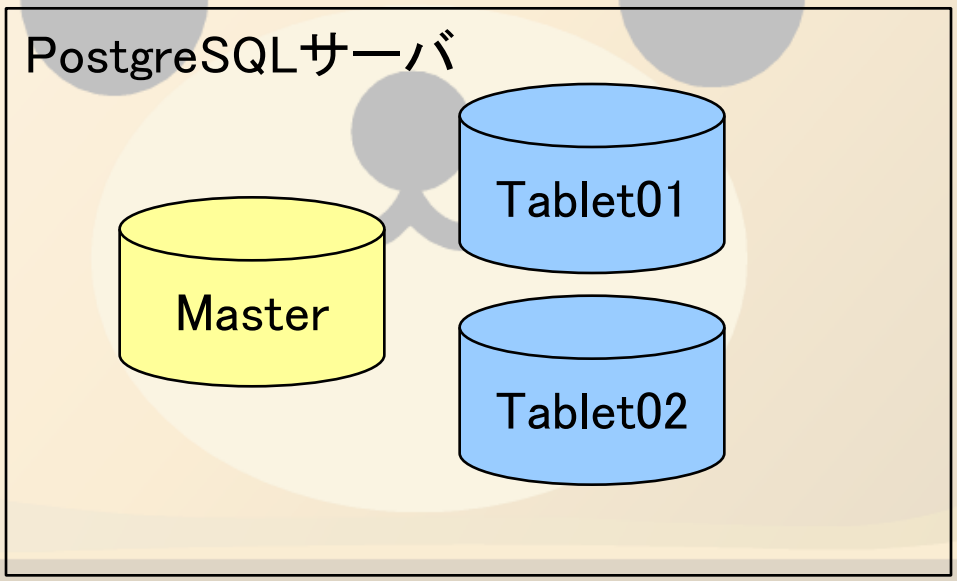
ちなみに、ORCAはLinux(Debian)で動きますよ。
ついでにいうと言語はCOBOLです。
あと、クライアントはJava版があるんでWindowsでもOKです。

全体構成について

サーバ側

1台のサーバにPostgreSQLデータベースを全部で3つ作成します。

Masterデータベース(RootTabletとMETADATATablet)×1
Tabletデータベース(実データが格納されるUserTablet)×2



全体構成について

クライアント側

DAOレイヤ

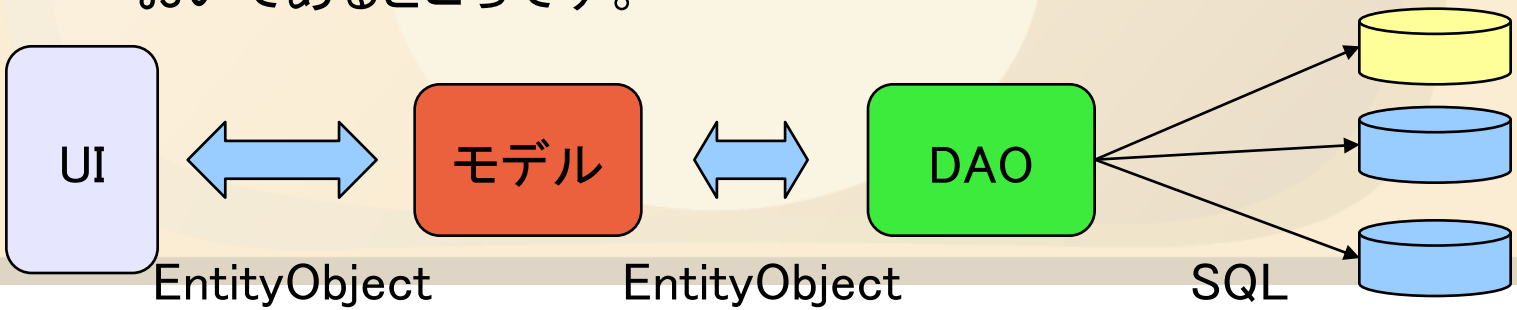
実際にデータベースとやり取りを行うためのロジックがおいてあるところです。今回は接続先のデータベースは3つ固定とします。

モデルレイヤ

データベースに格納したデータのエンティティモデルとCRUD用のロジックがおいてあるところです。

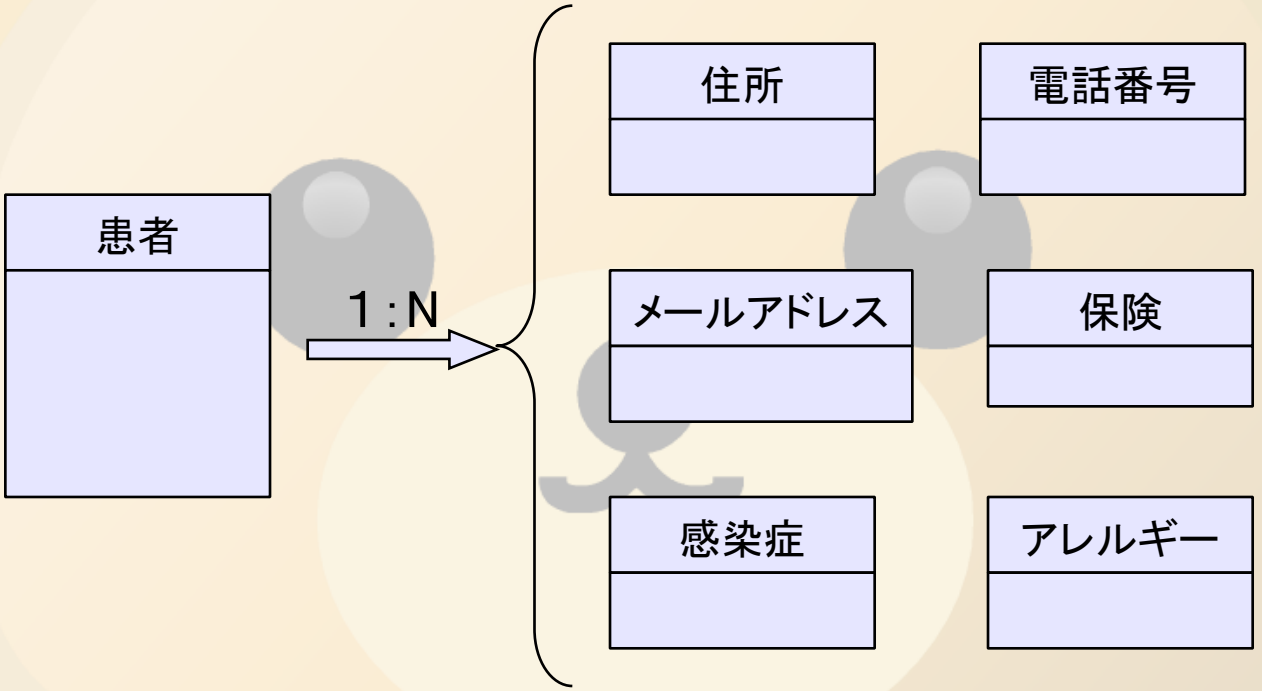
UIレイヤ

ユーザインターフェース用のフォームやテストコードなどがおいてあるところです。



エンティティモデル

今回のエンティティモデルは患者属性情報をつかいます。



テーブル構造1

Tabletテーブル

RowKey	text
ColumnName	text
ColumnIndex	int
SubColumnName	text
TimeStamp	timestamp
StartDate	date
EndDate	date
Value	text
UpdateDate	date
UpdateOperator	text
UpdateTerminal	text

Bigtableと異なるところ

1）期間のデータを扱うためタイムスタンプとは
別に開始日、終了日を加えています。
2）ColumnFamilyとしての子要素に対応するため、
ColumnIndexとSubColumnNameを追加しました。
PrimaryKeyは
RowKey+ColumnName
+ColumnIndex+SubColumnName
+TimeStamp
としました。
3）誰がいつどこから更新したか確認できるように、
更新日時、更新者、更新端末を追加しました。

テーブル構造2

METADATAテーブル

RowKey	text
TabletServerId	text
VersionNo	bigint

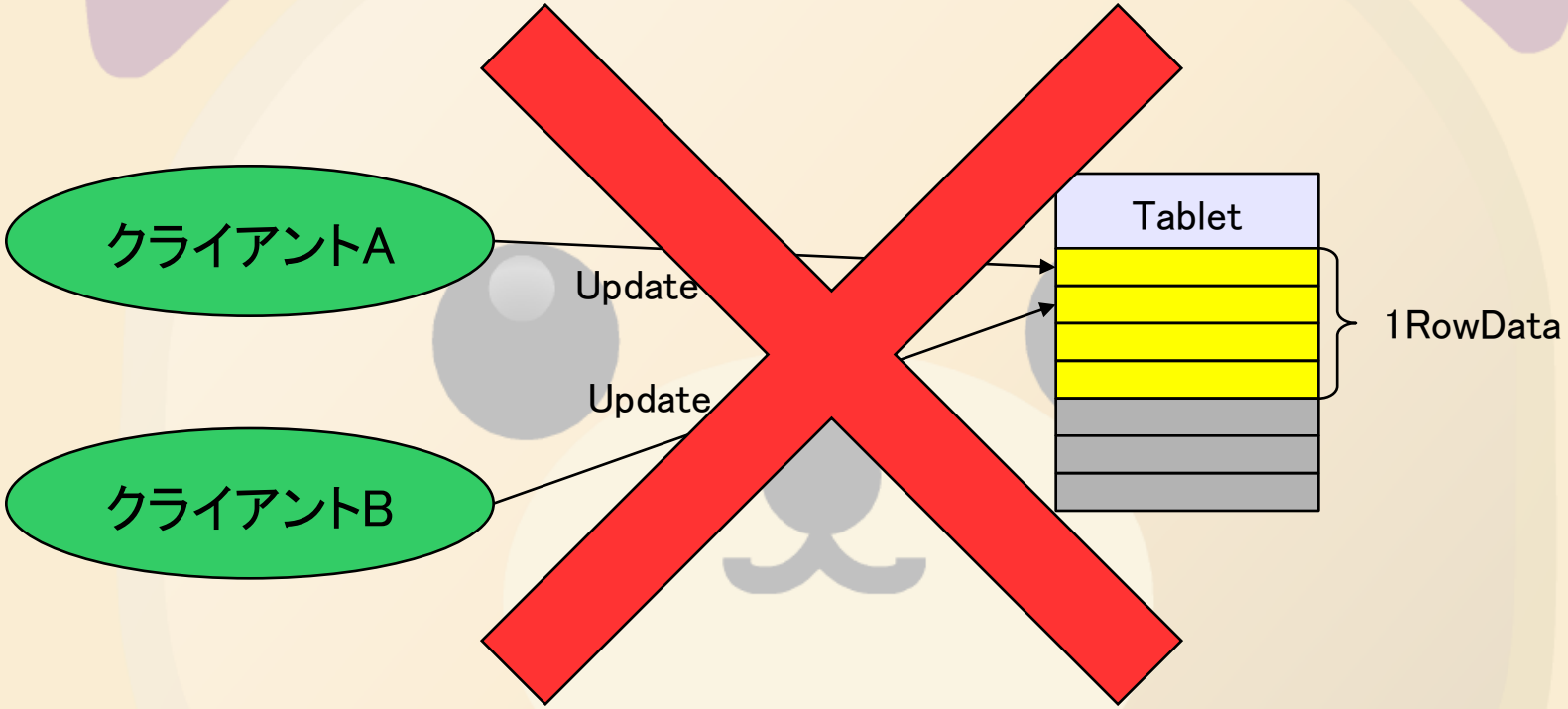
Bigtableと異なるところ

1）RootTabletとMETADATATabletは一緒にしました。
2）行キーごとに格納しているTabletサーバ
　の位置を保存します。
3）トランザクションのロック更新用のVersionNo
　を持っています。このカラムは更新ロック時
　にインクリメントしていきます。

エンティティとテーブルの対応

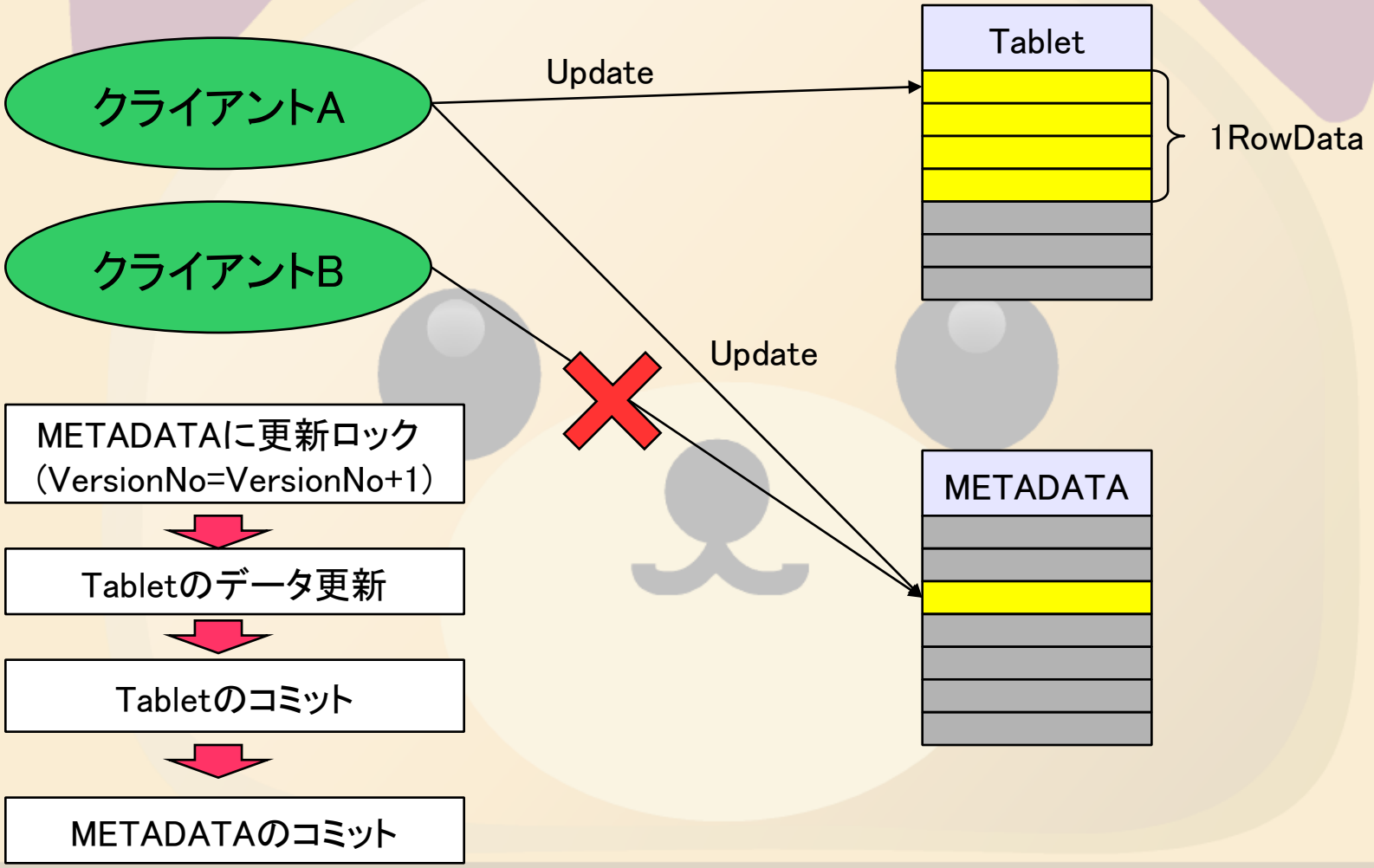
	RowKey	Column Name	Column Index	Sub ColumnName
患者のインスタンス	1対1			
患者の各プロパティ	1対1			
住所、電話番号、メールアドレス、保険、感染症、アレルギーのインスタンス	1対1			
住所、電話番号、メールアドレス、保険、感染症、アレルギーの各プロパティ	1対1			

トランザクションの実装



データベースが行単位のロック機構の場合、データの整合性が取れなくなる可能性がある！！

トランザクションの実装



新患登録

＜処理の概要＞

METADATAに管理情報を追加します。このとき、
どのTabletにデータを格納するかは各Tabletのデータ数で決定します。
今回は2サーバとなるので、交互にデータが登録されていく形となります。

患者情報オブジェクトからTablet更新用オブジェクトを
生成してDAOへ引き渡します。

DAOは更新オブジェクトからINSERT文を生成して
データベースに登録します。

患者情報取得

＜処理の概要＞

患者番号がRowKeyとなるのでMETADATAを患者番号で検索し、
該当キーがどのTabletにあるか取得します。

Tabletの位置を特定したら、そのTabletに対してRowKeyと対象日
で検索してデータを取得します。

ColumnName、ColumnIndex、SubColumnNameと患者情報オブジェクト
のプロパティをチェックしてオブジェクトにセットできるデータだけセットします。

患者情報オブジェクトを返します。

患者情報更新

＜処理の概要＞

患者番号でMETADATAを検索して該当キーが含まれているTablet
の位置を取得します。

更新対象のカラムデータを取得して、それぞれのカラムに対して
変更データのチェックと既存データの開始日終了日の調整を行います。

データ追加や更新が発生した場合はTablet更新情報オブジェクト
を生成してDAOへ引き渡します。

DAOは更新オブジェクトからINSERTまたはUPDATE文を生成して
データベースを更新します。

患者情報削除

＜処理の概要＞

患者番号でMETADATAを検索して該当キーが含まれているTablet
の位置を取得します。

Tabletの患者情報を削除するオブジェクトを生成して
DAOへ引き渡します。

DAOは更新オブジェクトからDELETE文を生成して
データベースから削除します。

METADATAから患者番号を削除します。

患者リスト取得

＜処理の概要＞

検索フィルタに格納されたColumnKeyと条件値で全Tabletに対して
RowKeyを取得するためのSQLを発行します。

各Tabletの結果をマージします。また、複数条件の場合に重複した
RowKeyがあったときは除外します。

マージされた結果をもとに患者情報取得メソッドを実行して患者リスト
を取得します。

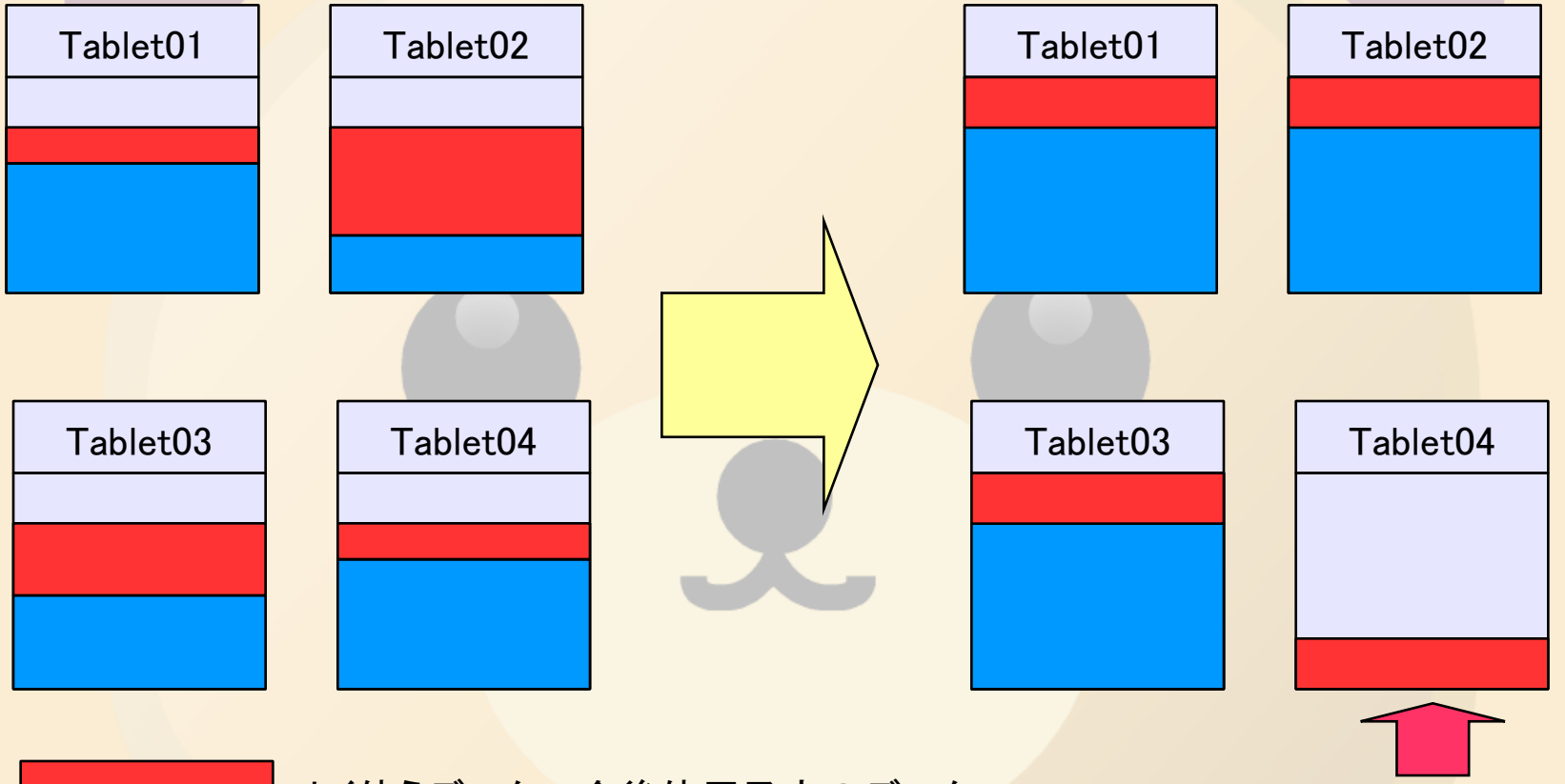
今後やってみたいこと

☆　データの再配置

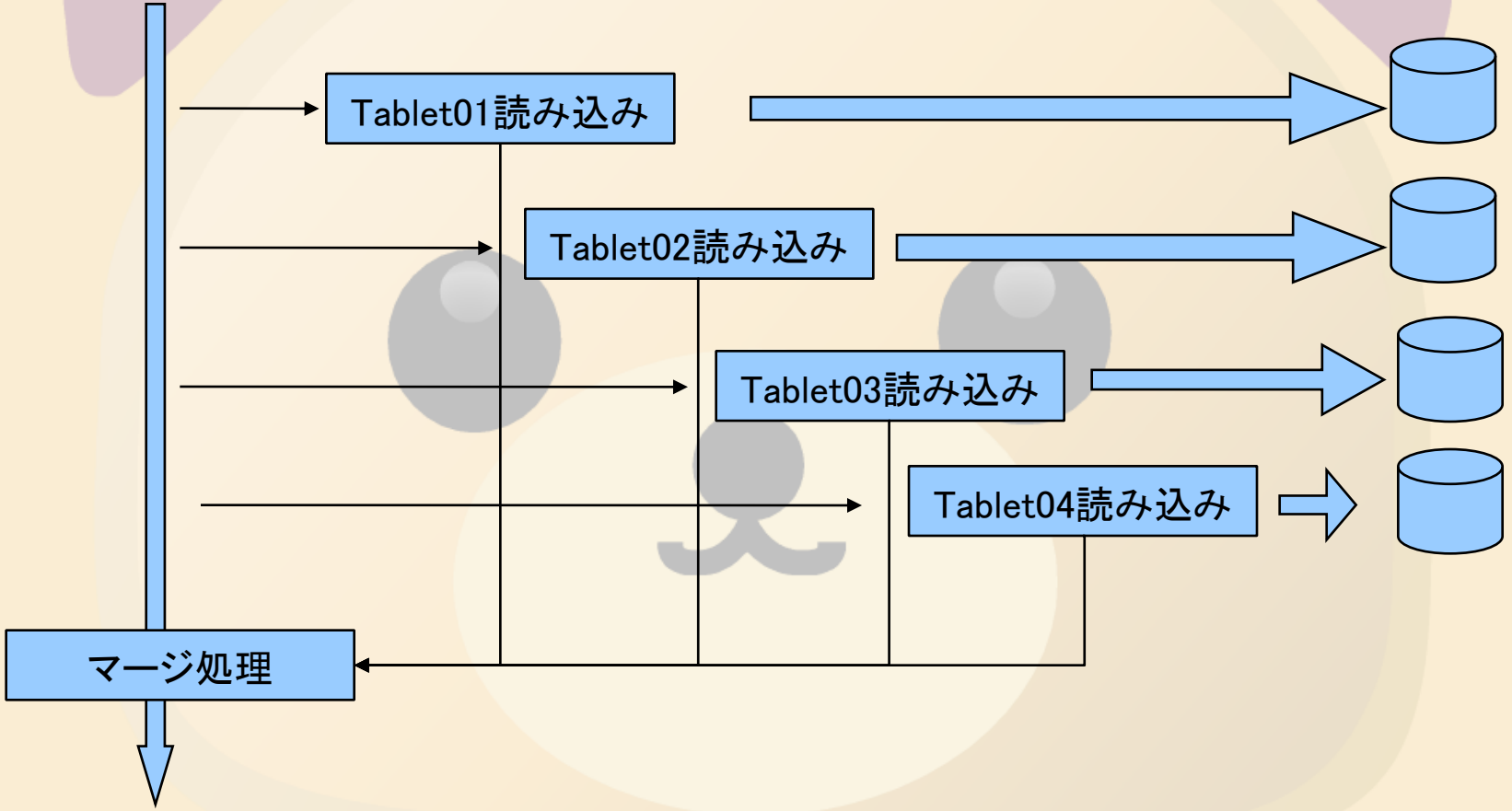
☆　マルチスレッド化

☆　読み込みプロパティの限定

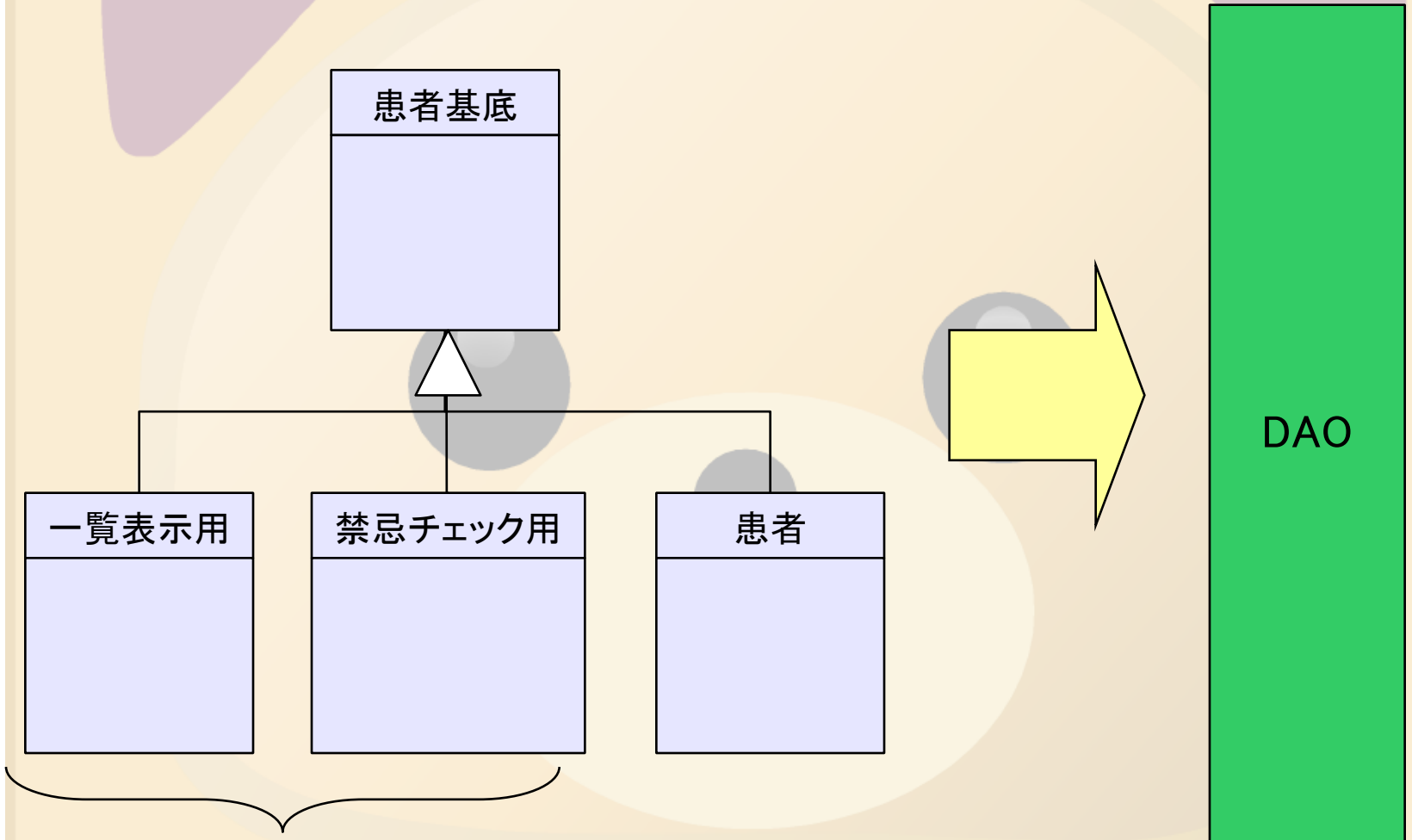
データの再配置


Tablet01
Tablet02
Tablet03
Tablet04
Tablet01
Tablet02
Tablet03
Tablet04
よく使うデータOr今後使用予定のデータ
今後あまり使用しないと思われるデータ
新規追加用

マルチスレッド化


Tablet01読み込み
Tablet02読み込み
Tablet03読み込み
Tablet04読み込み
マージ処理

読み込みプロパティ限定



という感じで
今後もちまちまとやっていきたいと思っています。

ご静聴ありがとうございました。