

Waha! Day 2023



実践！！ Waha! Transformer

～アイディア次第で色々デキル実装例～

2023年10月20日

KDDI株式会社

情報システム本部 宇南山清高

会社概要

会社名	KDDI株式会社
代表取締役社長	高橋誠
創業	1984年(昭和59年) 6月1日
事業内容	電気通信事業
資本金	1,418億円
従業員数	49,659名(連結ベース) ※2023年3月現在

Tomorrow, Together 

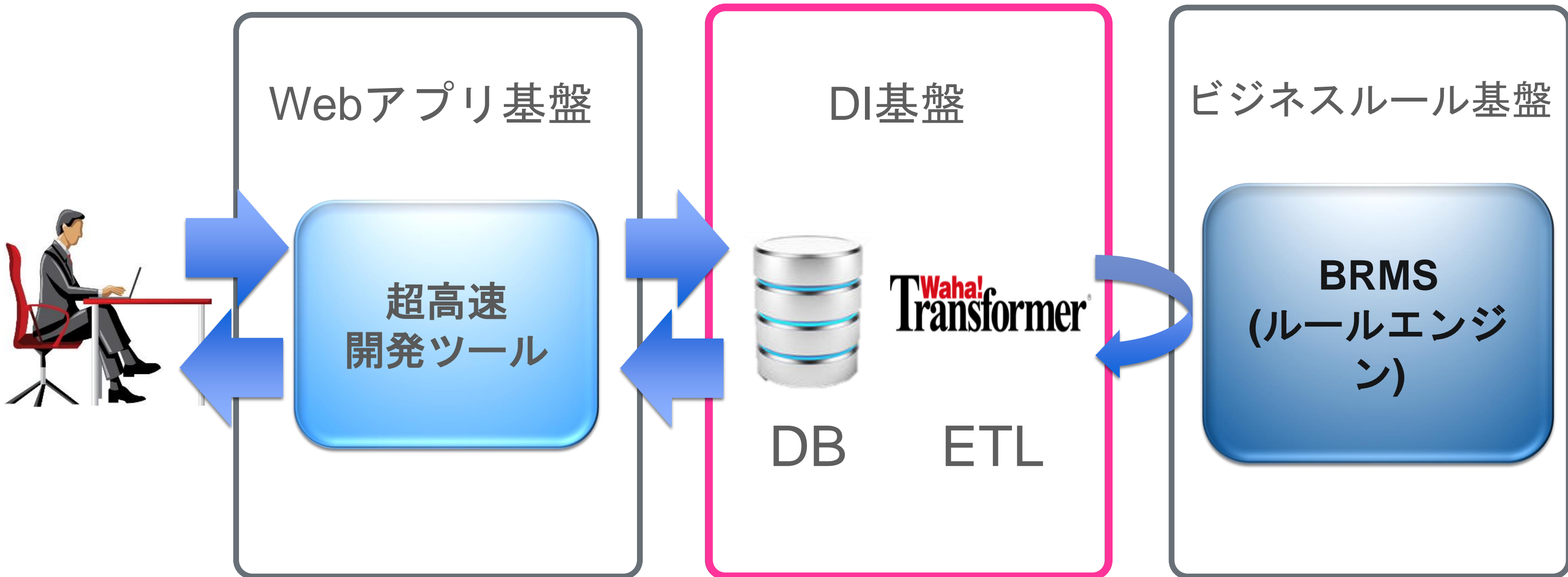
1 主なWaha!利用シーン

2 実装例の紹介

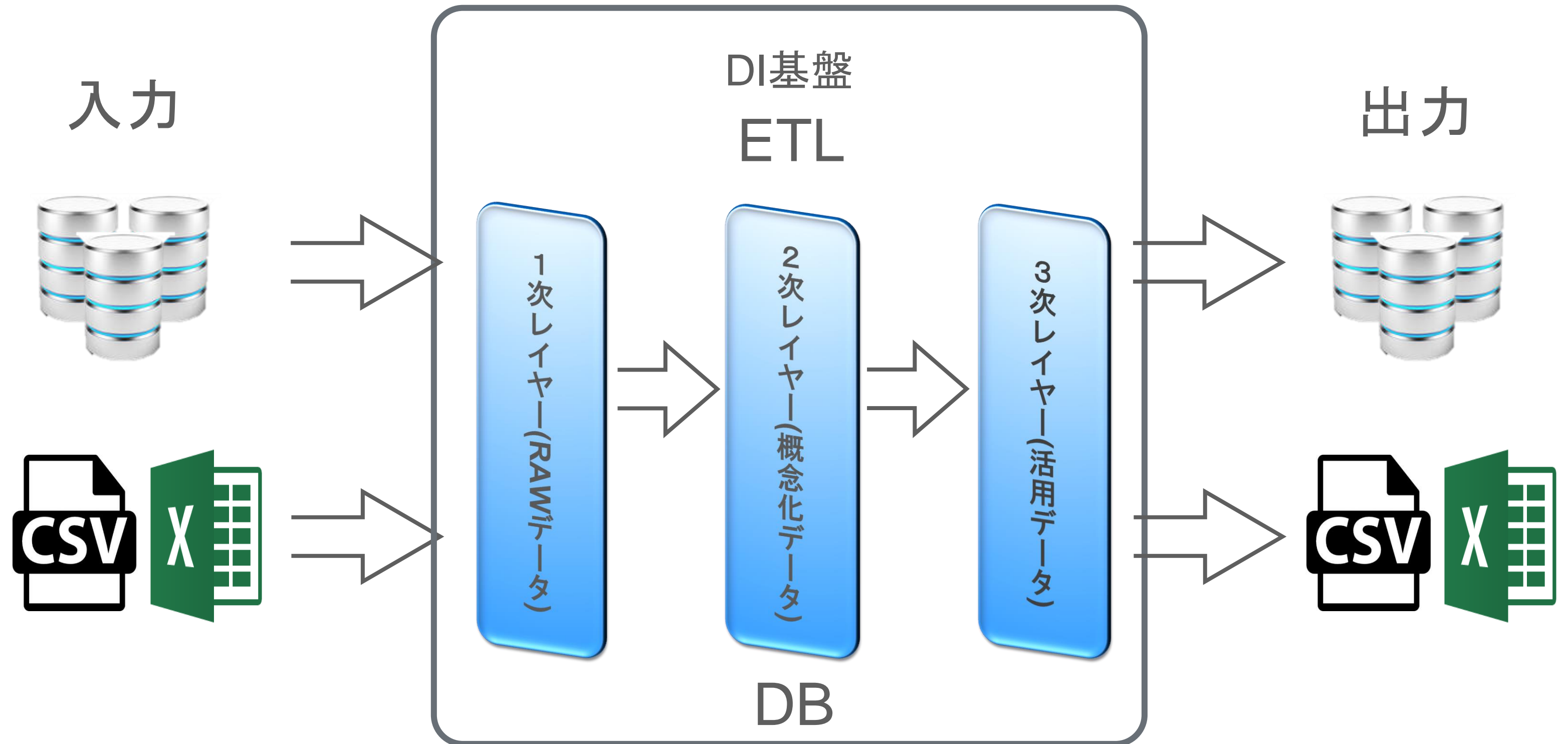
3 まとめ

主なWaha!利用シーン

業務IT基盤全体アーキテクチャ



データインテグレーション基盤



Waha!導入による効果

- 年間開発コスト約40%削減
コーディングレス化

- システム運用時間約45%削減
ドキュメントレス化

- 開発規模は3倍に拡大

処理・運用の見える化

四半

機能に

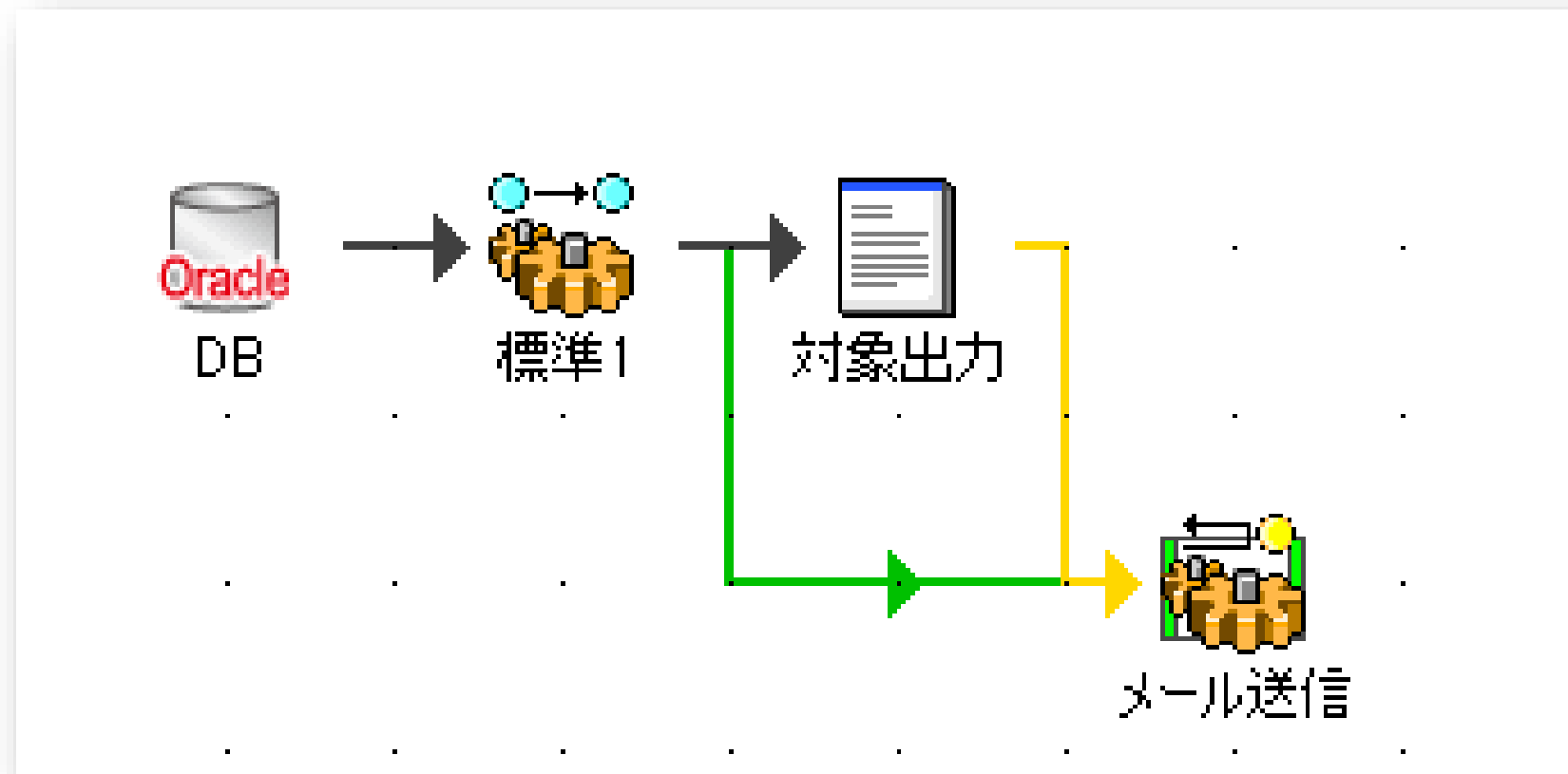
実装例の紹介

システムでエラーが発生したら
メール配信したい

例

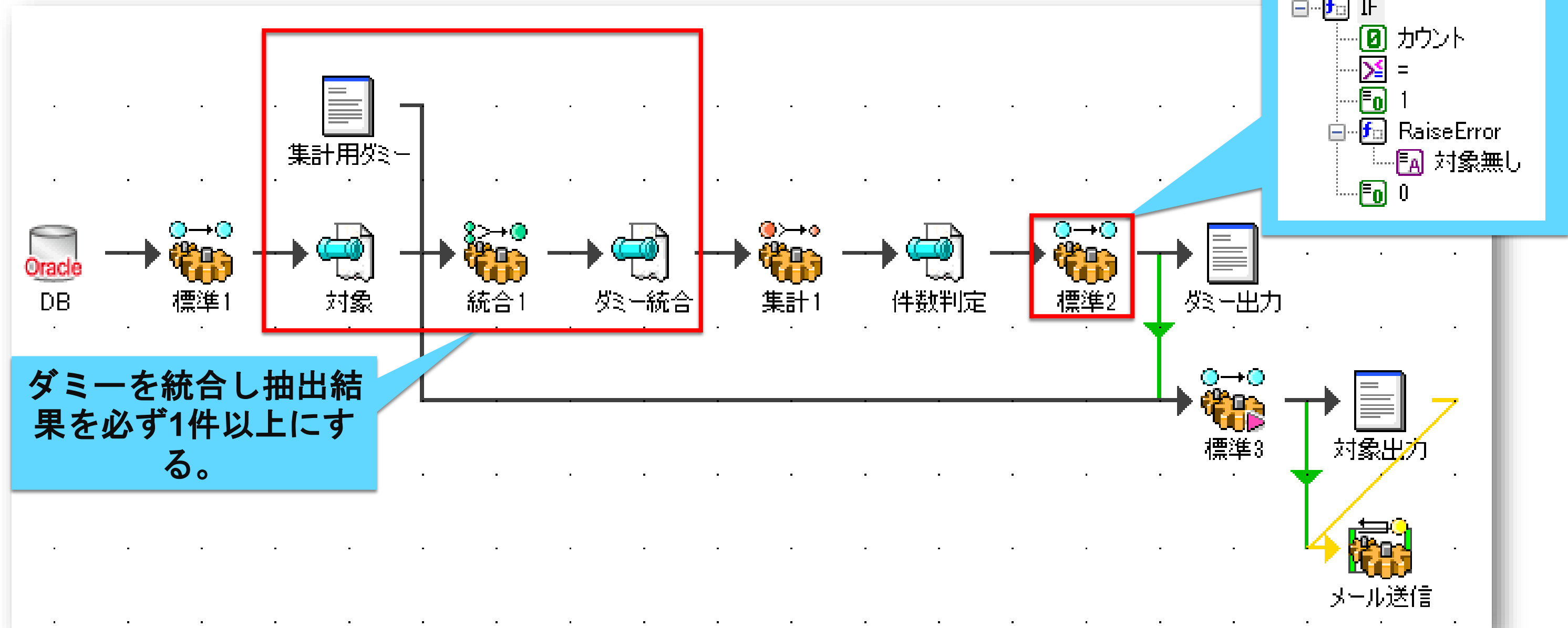
**DBのレコードを毎時検索し
検知対象のレコードが存在していたら
メール配信を行う。**

ダメな実装



これだと、対象が0件であってもメールが配信されてしまう。
毎分実行等のスケジュールにすると
メールの受信フォルダが大変なことに！！

こんな実装にしてみました。



件数集計と件数によるエラー判定(0件だったら異常終了)のロジックを追加する。
単純に集計すると0件だった場合に、エラー判定が正しく実行されず
JOBが正常完了し、必ずメールが配信されてしまう。



ダミーレコードを追加して集計する

RaiseErrorで処理を終了する

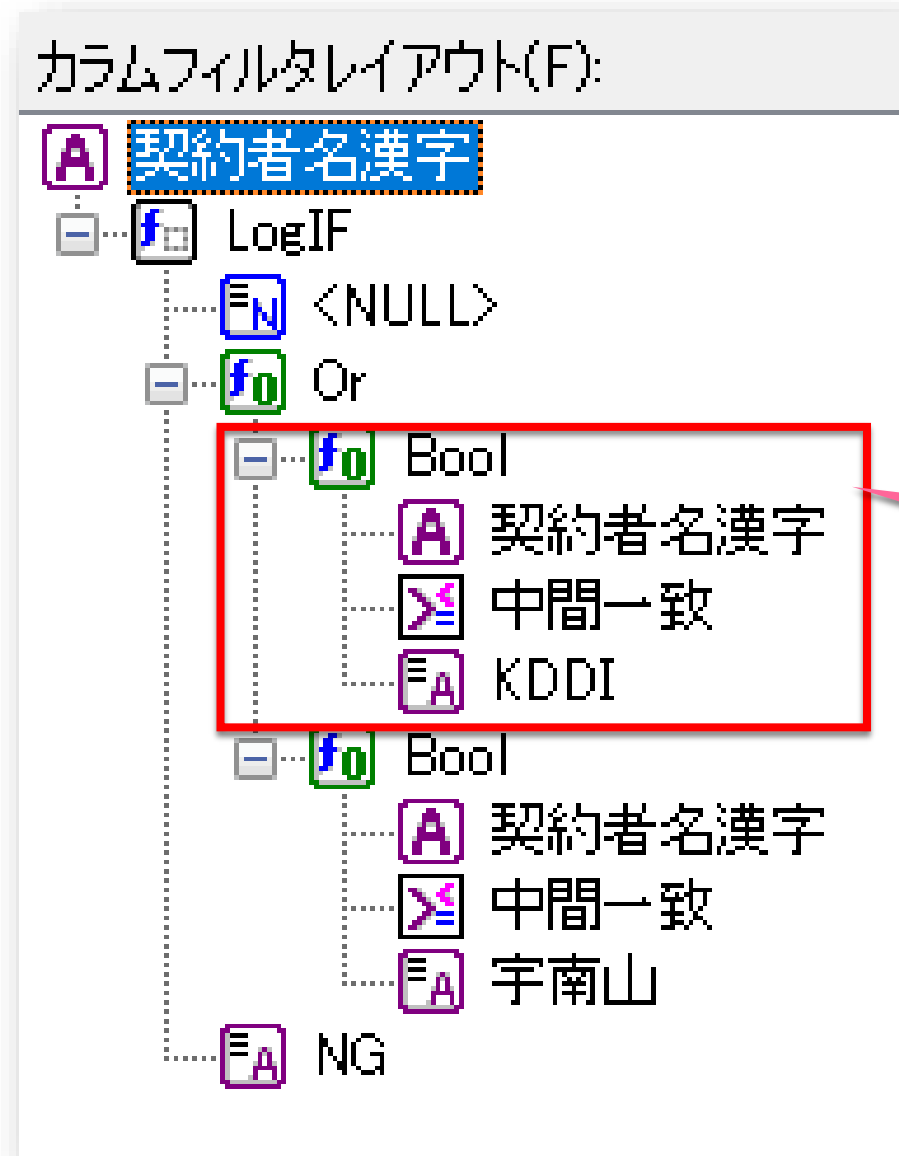
禁則文字をチェックしたい

例

DBのレコードから対象の文字列を抽出する。

**他のシステムへデータ連携する場合の
チェック等**

ダメな実装



対象文字列の個数分だけ
OR条件を追加する必要があり面倒

これでも要件は満たせる。
しかし、対象文字列が多い場合に実装が大変なことに！！

こんな実装にしてみました。

カラムフィルタレイアウト(F):

A 契約者名漢字

f IF

fA MatchRE

A 契約者名漢字

A KDDI | 宇南山 | ユニタ

?0 未設定(数値型)

=

N <NULL>

N <NULL>

fA MatchRE

A 契約者名漢字

A KDDI | 宇南山 | ユニタ

?0 未設定(数値型)

正規表現にマッチした文字列を返す。

| で区切られた文字列が存在した場合に一致としてその文字列を返す。

一致した場合に同じ関数を入れることで、一致した文字列を出力できる。

正規表現を用いることで、対象文字列が多くても簡単に実装することができる。



正規表現と関数を組み合わせる

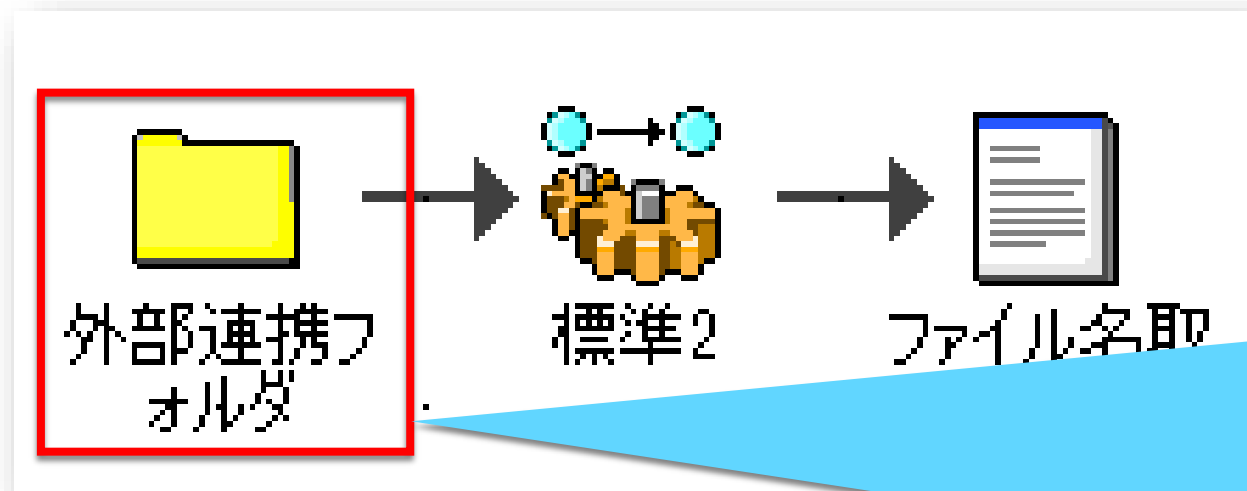
ファイルが取り込まれていなかったら連絡がほしい。

例

何らかの理由でファイルトリガが動かなかったことを検知する。(IN)

外部に連携するファイルが、連携先に取り込まれていないことを検知する。(OUT)

こんな実装にしてみました。



ファイル一覧入力接続情報を使う

ファイル一覧入力接続情報のプロパティ

名前(N): 外部連携

コメント(D):

更新日時: 2021/09/21 17:30:17

プロパティ(P):

プロパティ名	設定値
パス	C:\%XXX%\in%X連携ファイル

ファイルトリガの監視フォルダや，外部へIFするファイルのパスにあるファイル名一覧を取得し，1ファイルでもあった場合に通知する。



ファイル一覧入力接続情報を使う

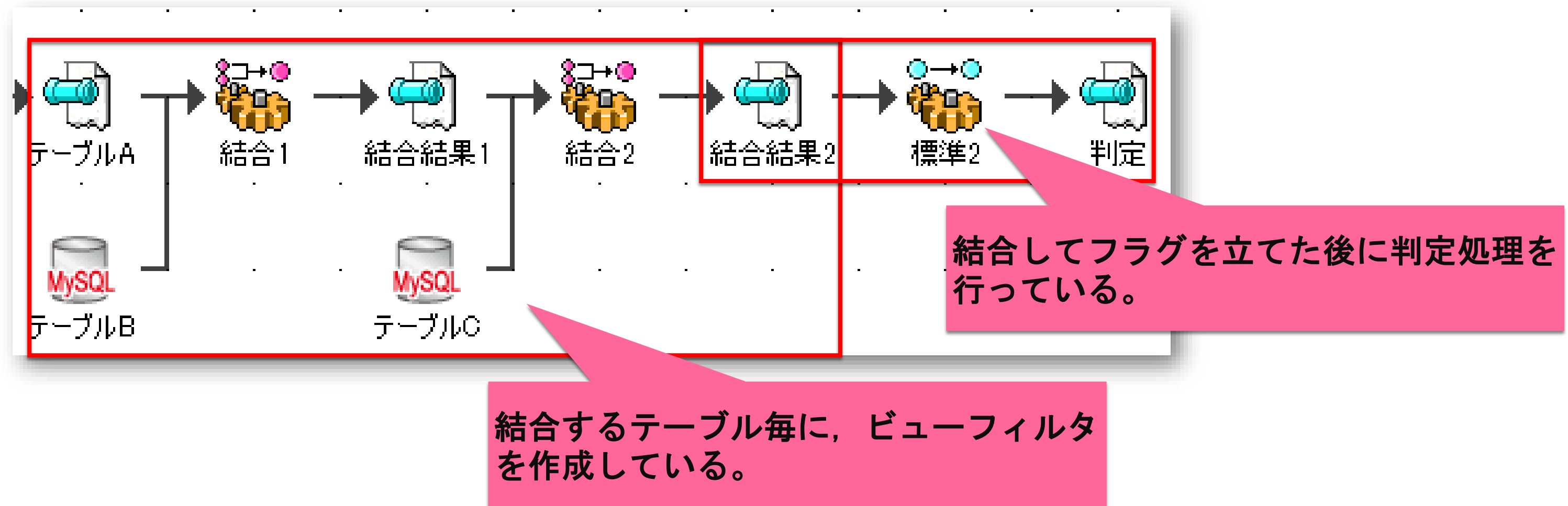
テーブルを結合してその結果で
判定をしたい

例

テーブルAとBとCを比較して，フラグを立てる。

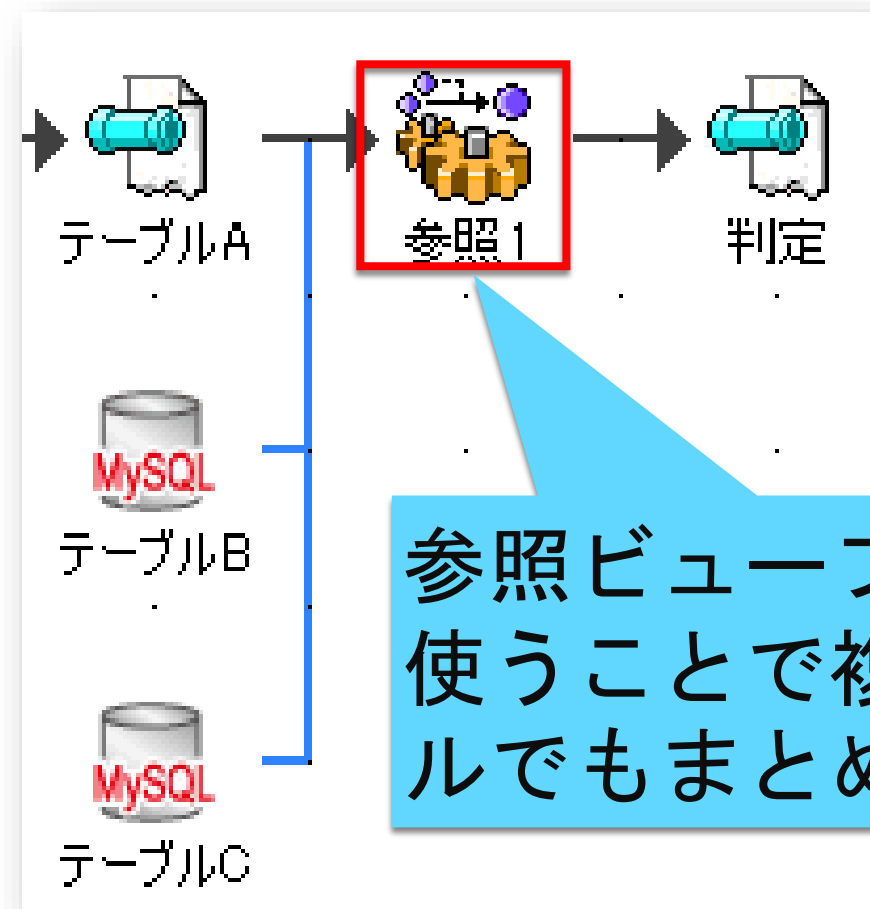
更に，同時に複数の結合条件でandやorの比較を行う。

よくある実装



これでも要件は満たせるが、結合するテーブルや判定条件が増え
ると、伴ってビューフィルタが増えてしまう。

こんな実装にしてみました。

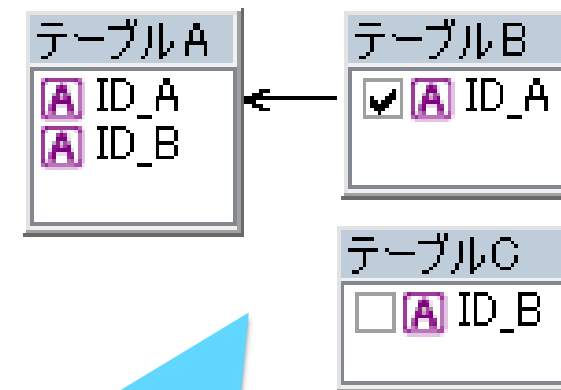


参照ビューフィルタを使うことで複数テーブルでもまとめられる

参照条件一覧(L):

	参照条件名	参照ビュー名
1	参照条件1	テーブルB テーブルC
2	参照条件2	テーブルB テーブルC

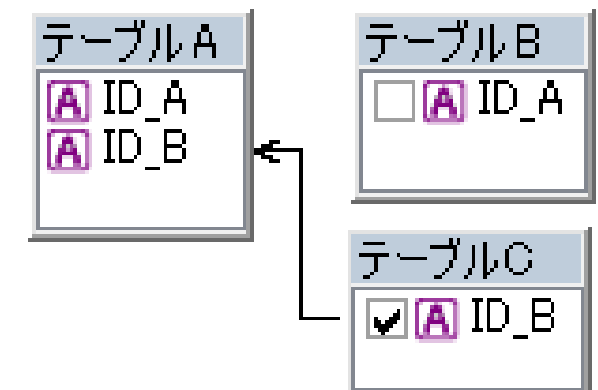
プロパティ



参照条件一覧(L):

	参照条件名	参照ビュー名
1	参照条件1	テーブルB テーブルC
2	参照条件2	テーブルB テーブルC

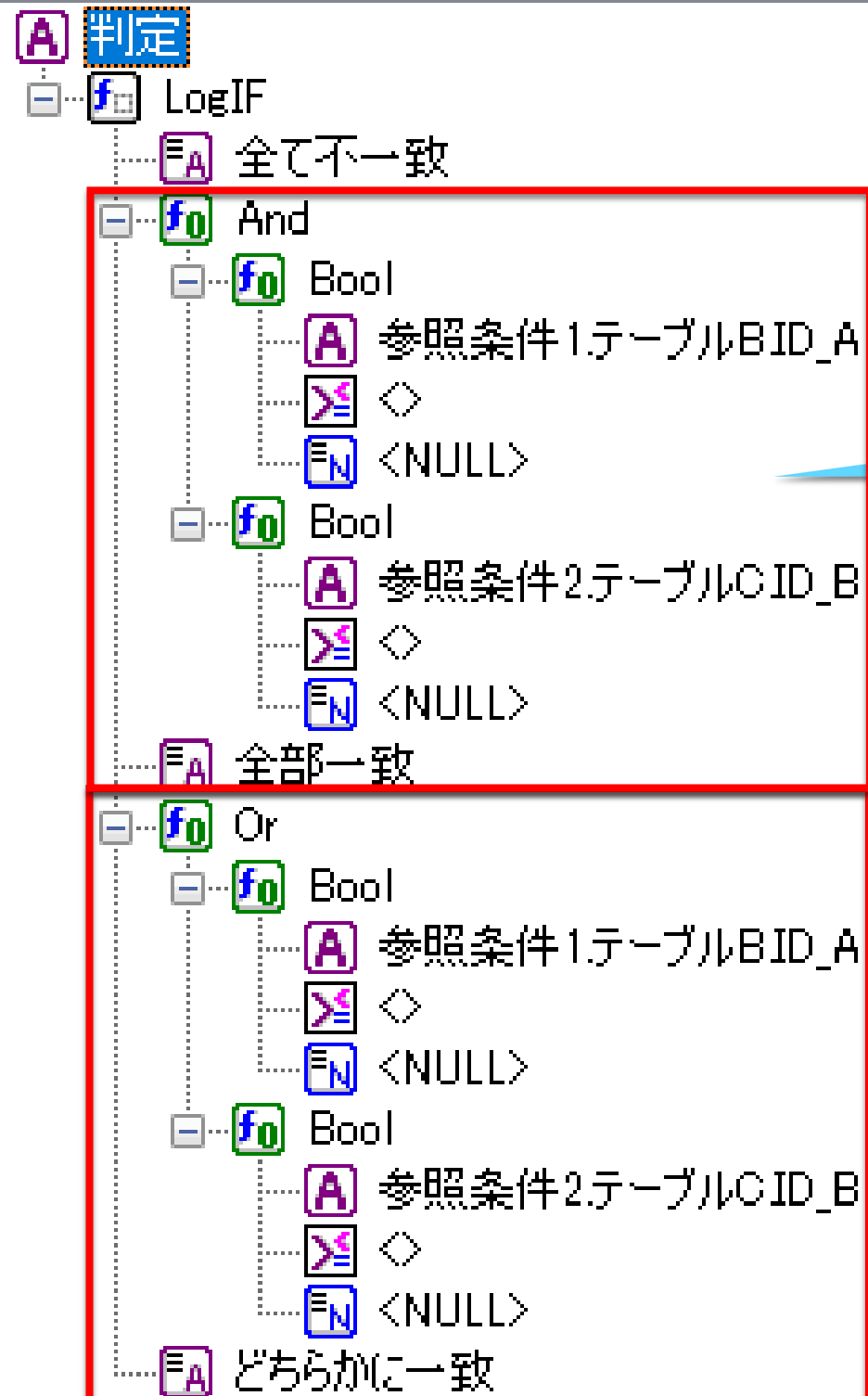
プロパティ



参照条件を追加することで複数のテーブルやカラムで比較できる

こんな実装にしてみました。(続き)

カラムフィルタレイアウト(F):



様々な条件で判定を行える。

- ・ 比較テーブルが増えてもビューフィルタが増えない。
- ・ 参照ビューフィルタ内で判定処理まで行える。
- ・ 複雑な判定も柔軟に行える。



参照ビューフィルタは結合ビューフィルタのような使い方ができる

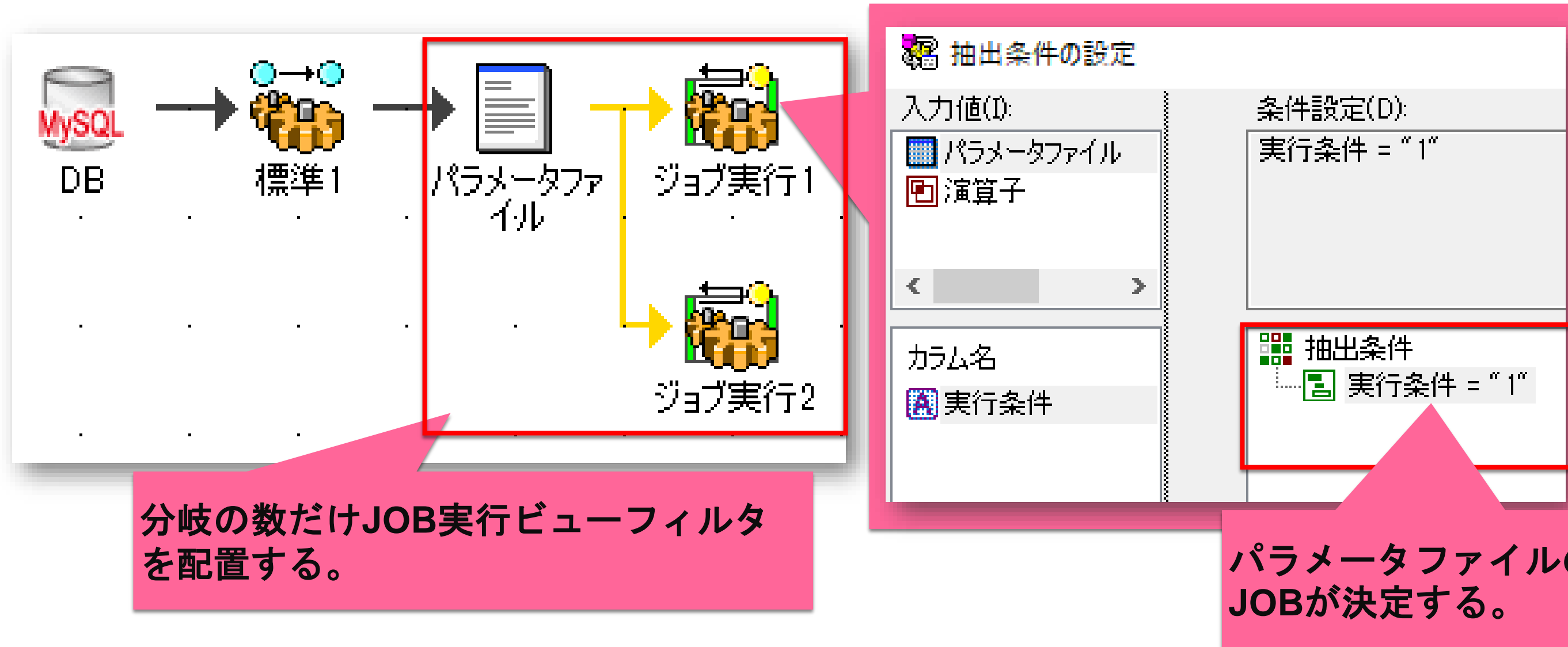
参照ビューフィルタで複数結合条件を設定できる

条件に従ってJOB実行ビュー
フィルタで実行するJOBを変え
たい。

例

何らかの判定を行って，その結果に従って次に実行するJOBを決定する。

ダメではないけどイマイちな実装



これでも要件は満たせるが、分岐の数だけJOB実行ビューフィルタを配置する必要がある。

こんな実装にしてみました。



トリガファイル用のパラメータファイルを作成。
ファイル名や出力先, 等を格納する。

条件分岐で実行するJOBが監視する先。

受け取ったパラメータでトリガファイルを作成する。

ビューのプロパティ

ビュー名(N): トリガファイル

ファイル名(E): `$EINFOLDER$$$ETRGFILENAMES$`

モード(M): Read/Write

文字コード(C): シフトJIS

カラム設定(L):

	カラム名	型	精度	小数以下
1	param	文字	255	

こうすることで, 元のJOBは変更せずに条件増減ができる。



次のJOBはJOB実行ビューフィルタで直接実行しない

ファイルトリガで次のJOBを実行する

特殊な判定をしたい

例

アプリに1日1回ログインするとポイントが貰えるとする。

更に毎日続けてログインするとボーナスポイントが付与される。

今回はそのボーナスポイントを算出する。

やりたいこと

日付	ログインしたか	通常ポイント	ボーナスポイント
2023/9/1	○	1	0
2023/9/2	○	1	1
2023/9/3		0	0
2023/9/4	○	1	0
2023/9/5	○	1	1
2023/9/6		0	0
2023/9/7	○	1	0
2023/9/8	○	1	1
2023/9/9	○	1	1

ここを算出する

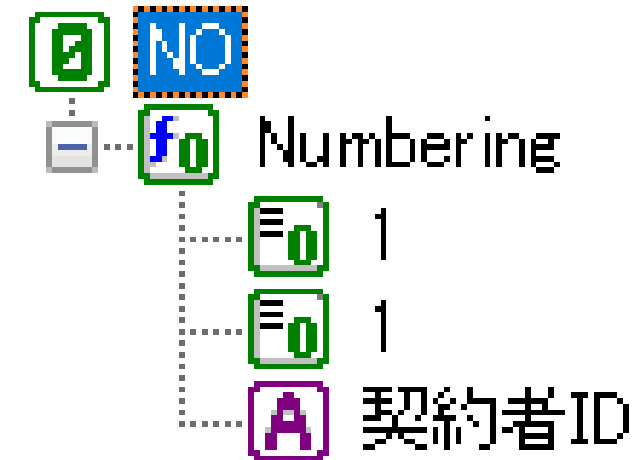
こんな実装にしてみました。

カラム設定(L):

	カラム名	型
1	NO	数値
2	契約者ID	文字
3	ボーナスポイント	数値
4	ログイン日	日付
5	前回ログイン日	日付

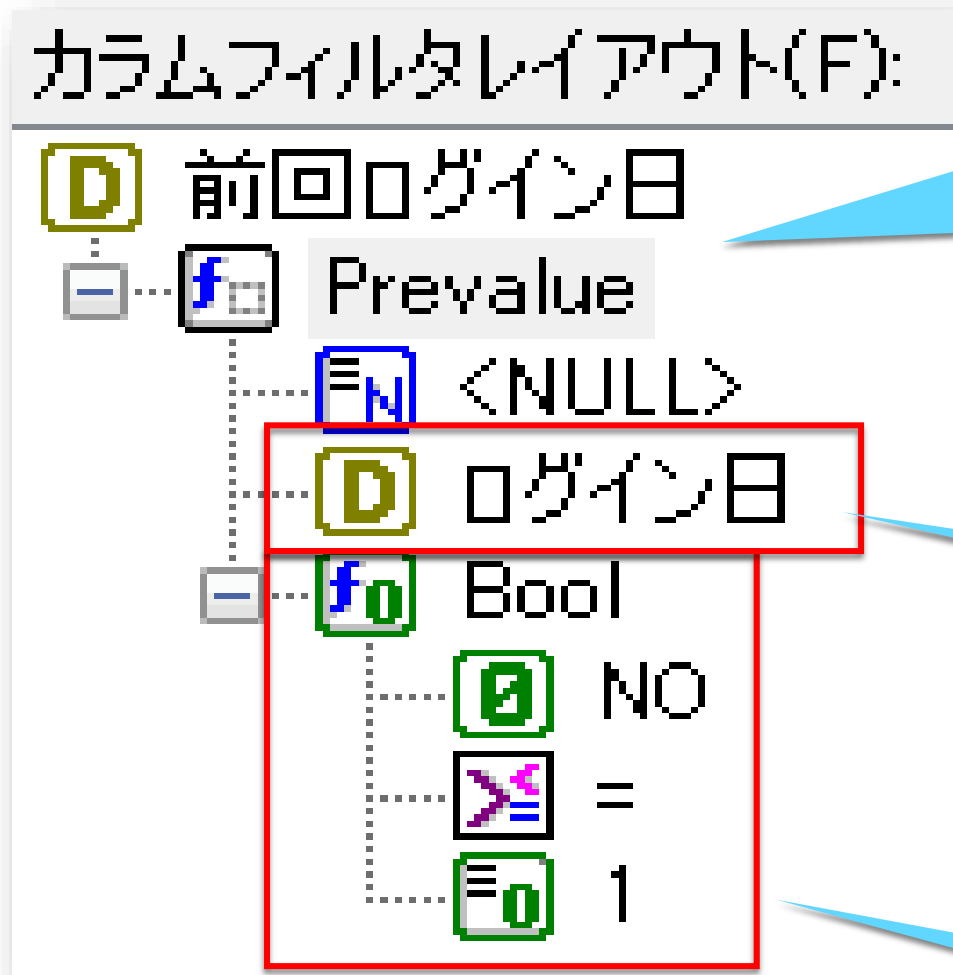
データイメージ
ログインする度にレコードが作成される

カラムフィルタレイアウト(F):



契約者ID毎に連番を付与する。
(事前にソートを忘れずに)

こんな実装にしてみました。(続き)



Prevalue関数を使ってログイン日の前日日付を取得する。(リセット条件に一致したらリセット値に戻し、一致しなかったら前回実行時の値を取得)

前回実行時の値なので1レコード前のログイン日(前回ログイン日)が取得できる。

リセット条件
契約者IDが変わったらリセットする。

こんな実装にしてみました。(続き)

カラムフィルタレイアウト(F):

ボーナスポイント

IF

-1

=

DateDiff

前回ログイン日

ログイン日

3

1

0

ログイン日と前回ログイン日を比較してポイントを付与する。

前回ログイン日から今回ログイン日を引く

結果-1(つまり前日)だったら1ポイント付与, そうでなければ0ポイント。

ログイン日と前回ログイン日を1レコードに同時に持たせることで連続した場合に・・・という実装ができました。



連番を付与してリセット条件にする

Prevalue関数で前回ログイン日を取得する

まとめ





メール配信等，通知系は埋もれさせないことが重要

- 無駄に通知が多いと形骸化してしまう
- 受取り側が必要な情報のみを配信する

工夫次第で見易さや，開発コストを削減できる

- イニシャル/ランニングのコストバランスを

ご清聴ありがとうございました。

Tomorrow, Together

KDDI