第5日午後

製造業ITマイスター指導者育成プログラム
研修テキスト　実習用教材（第5日）
システム構築技術の習得2
（データ分析）

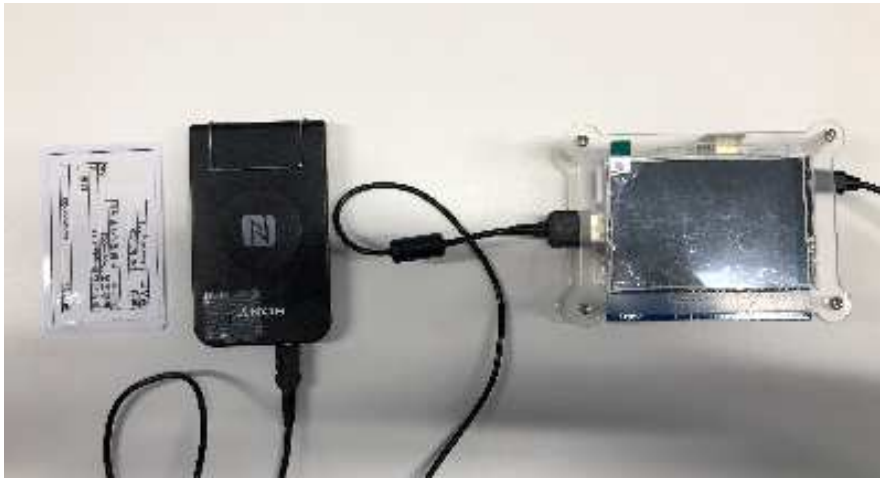
製造業ITマイスター研修教材一覧

日	テーマ		教材
1	製造業IT導入ワークショップ	午前	IoTとシステムの基礎
		午後	製造業IT導入ワークショップ
2	高度IT実装技術の習得１	午前	IoTによるシステム開発入門
		午後	高度IT実装技術の習得１（ラズパイ＋見える化実習）
3	高度IT実装技術の習得２	午前	IoTによる生産管理入門
		午後	高度IT実装技術の習得２（IoTセンサー実装実習）
4	システム構築技術の習得１	午前	IoTによる在庫管理入門
		午後	システム構築技術の習得１（業務システムの基本パターン）
5	システム構築技術の習得２	午前	IoTによるデータ分析入門
		午後	システム構築技術の習得２（データ分析）
6	PBL１（事例企業調査）	午前	事例企業調査
		午前	事例企業の課題モデル化実習
7	PBL２ （課題の設定と解決策の提案）	午後	システム構築の実際
		午後	システム構築実習（１）課題の設定と解決策の提案
8	高度IT実装技術の適用	午前	IT経営の実践方法
		午後	システム構築実習（２）高度IT実装技術の適用
9	システム構築技術の適用	午前	情報システムセキュリティ基礎
			知財とオープン＆クローズ戦略
		午後	システム構築実習（３）システム構築技術の適用
10	筆記試験および成果発表会	午前	個人と組織の発展に繋がるキャリアデザイン講座
			（筆記試験）
		午後	（成果発表会）

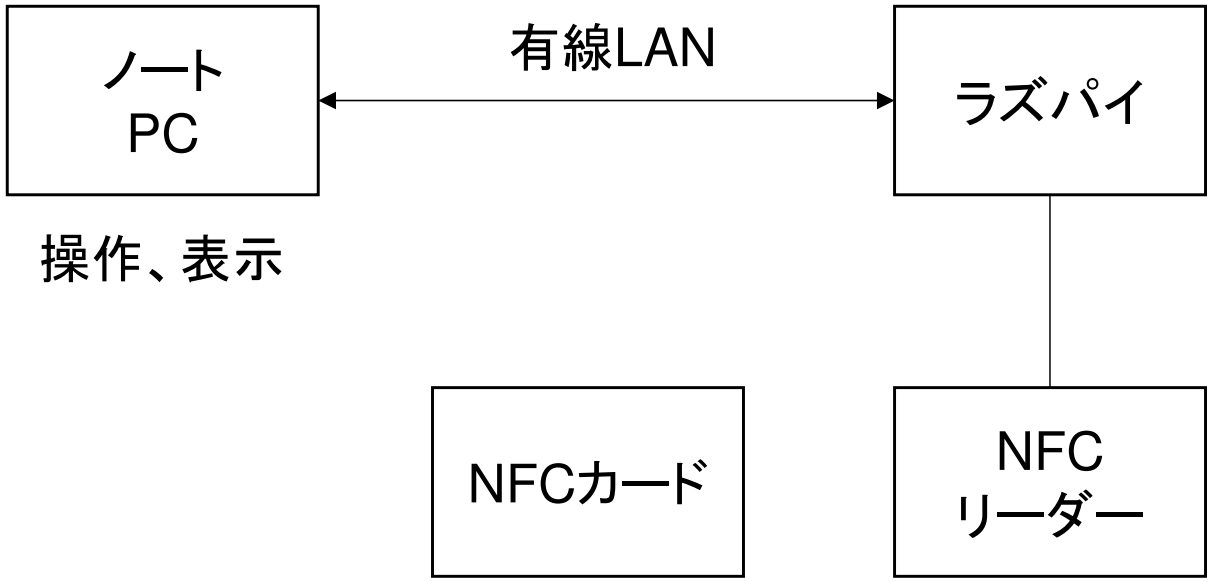
実習の内容(5日目後半)

Node-REDを使って
進捗管理システムを
作ってみよう！

最終形



システム構成



前半5日間の進め方

午後の実習

■1日目　実習のための環境設定 ⇨ 課題発見ワークショップ

■2日目　デバイス信号のイン/アウト ⇨ センサデータの見える化

■3日目　メールとWebサーバ利活用 ⇨ 人感センサとカメラの利用

■4日目　業務システムの基本パターン ⇨ バーコードリーダとNFC

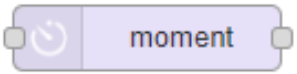
■5日目　データ分析続き ⇨ 工程進捗管理ボード

もくじ

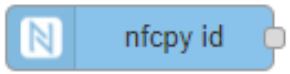
使用するノード

キーとなるノード一覧

- 時刻取得ノード



- RFID読み込みノード



- テンプレートノード

- データベースSQLiteノード

- Dashboardテンプレートノード

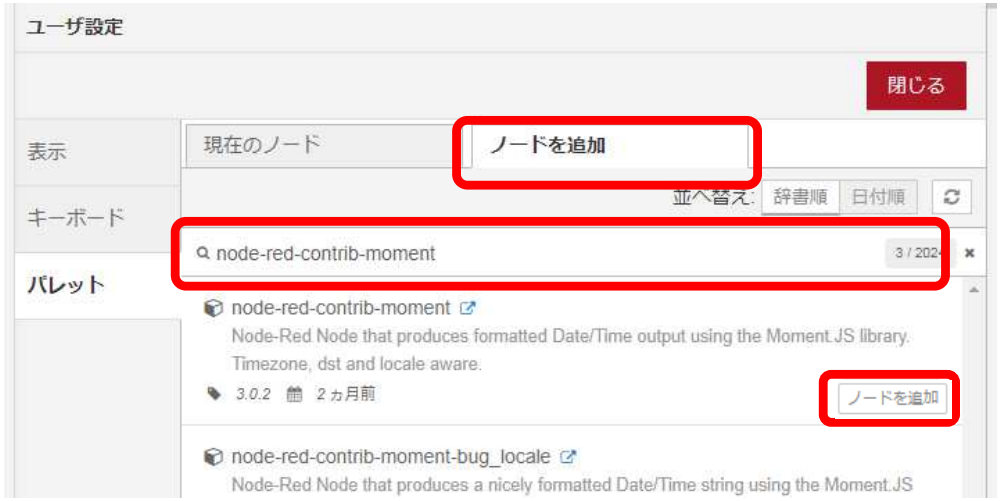
フロー追加

フローは、ワークスペースの1ページに相当するひと塊を指します。
フローの追加
新しいフローを追加するには右上の＋をクリックします。

時刻ノード追加

時刻を取得するノードは標準では入っていないため、ノード追加する必要があります。

右上の「パレットの管理」→「ノードを追加」→「node-red-contrib-moment」と入力
「ノード追加」を押す。



NFCノード追加

Node-REDで使用できるNFCリーダーのノードは、元々Pythonで作られたライブラリを
Node-REDで活用できる形に変換されたものです。

NFCリーダーのノードを使用する場合は、まずpipをインストールする必要があります。

pipとは・・・Pythonのパッケージを管理するためのツール
パッケージ：公式が配布もしくはサードパーティが配布するもの。

＜作業＞

1. sudo apt-get install python-usb python-pip –y
2. sudo pip install -U nfcpy-id-reader
3. cat << EOF | sudo tee /etc/udev/rules.d/nfcdev.rules
   SUBSYSTEM==“usb”, ACTION==“add”, ATTRS{idVendor}==“054c”,
   ATTRS{idProduct}==“06c3”, GROUP=“plugdev” EOF
4. sudo reboot
5. パレット管理より「node-red-contrib-nfcpy-id」を検索し「ノードを追加をクリック」

※3は一般ユーザーでNode-REDを立ち上げたときにも実行できるようにする方法。

NFCノード追加

NFCリーダーのノードは標準では入っていないため、ノード追加する必要があります。

右上の「パレットの管理」→「ノードを追加」→「node-red-contrib-nfcpy」と入力
「node-red-contrib-nfcpy-id」の「ノード追加」を押す。

SQLiteノード追加

取得した時間データなどを格納するための箱として、SQLiteを使用します。

右上の「パレットの管理」→「ノードを追加」→「node-red-node-sqlite」と入力
「node-red-node-sqlite」の「ノード追加」を押す。

Dashboard追加

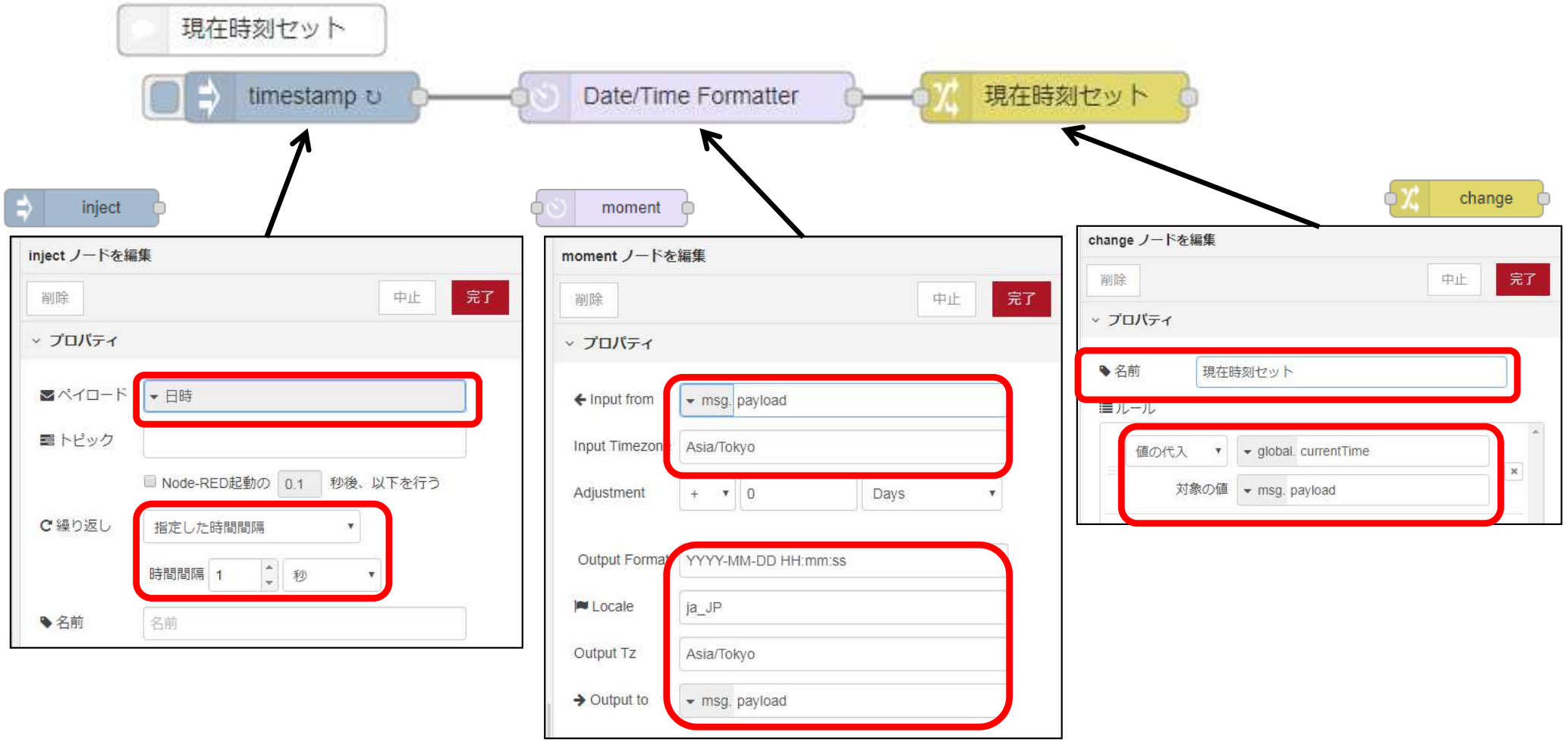
Node-REDで結果を画面出力する場合には、Dashboardを使用します。

右上の「パレットの管理」→「ノードを追加」→「node-red-dashboard」と入力
「node-red-dashboard」の「ノード追加」を押す。

もくじ

現在時刻取得

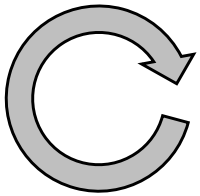
現在時刻を取得し、変数に格納します。



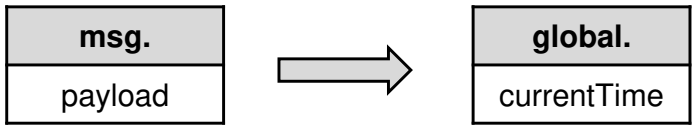
※Point　現在時刻セット　・・・コメント　comment

現在時刻取得（解説）

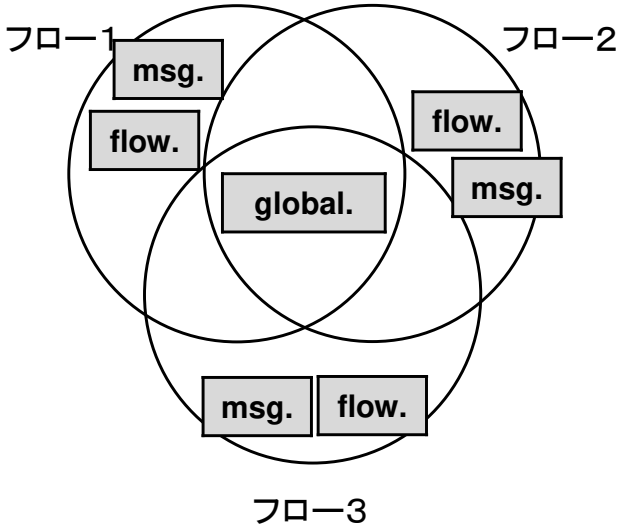
現在時刻を取得し、変数に格納します。



1秒周期



※　補足

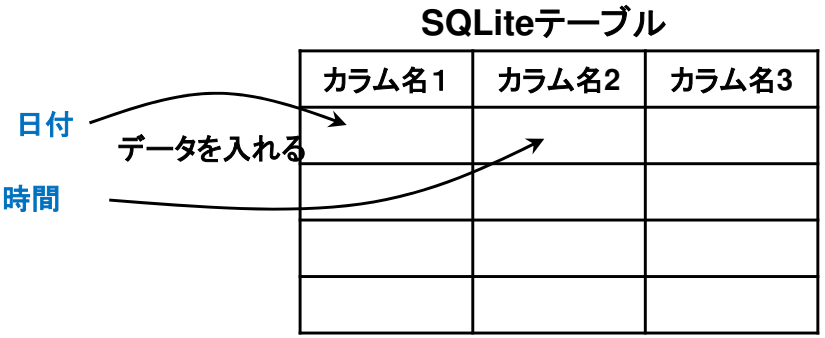


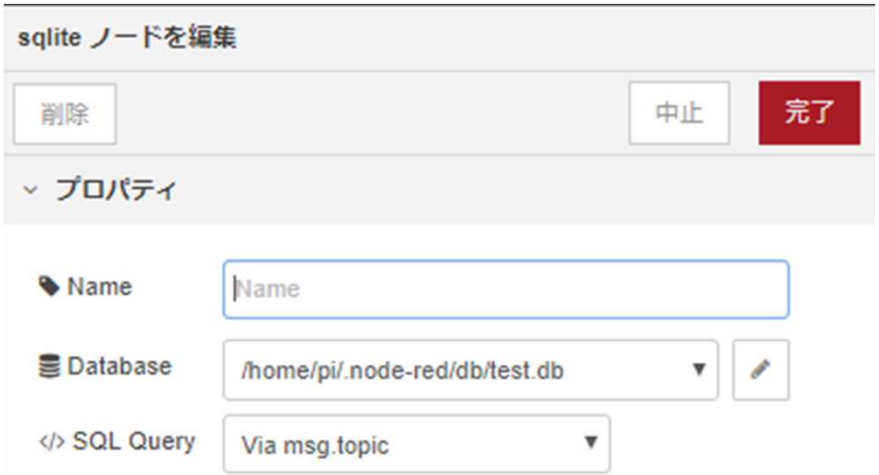
msg.payloadに入れたときに、格納形式を指定している。

もくじ

SQLiteデータベースについて

データベースとは、データを一時的に格納する箱のようなものです。
SQLiteは、簡単に使用することができるデータベースの種類。
SQLiteを使用する場合は、テーブルを作成する必要があります。
クエリ文という命令をすることで、データベースを操作することができます。





クエリ文例

テーブル作成

```
CREATE TABLE
    テーブル名(
        カラム名1,
        カラム名2,
        ・・・・
    );
```

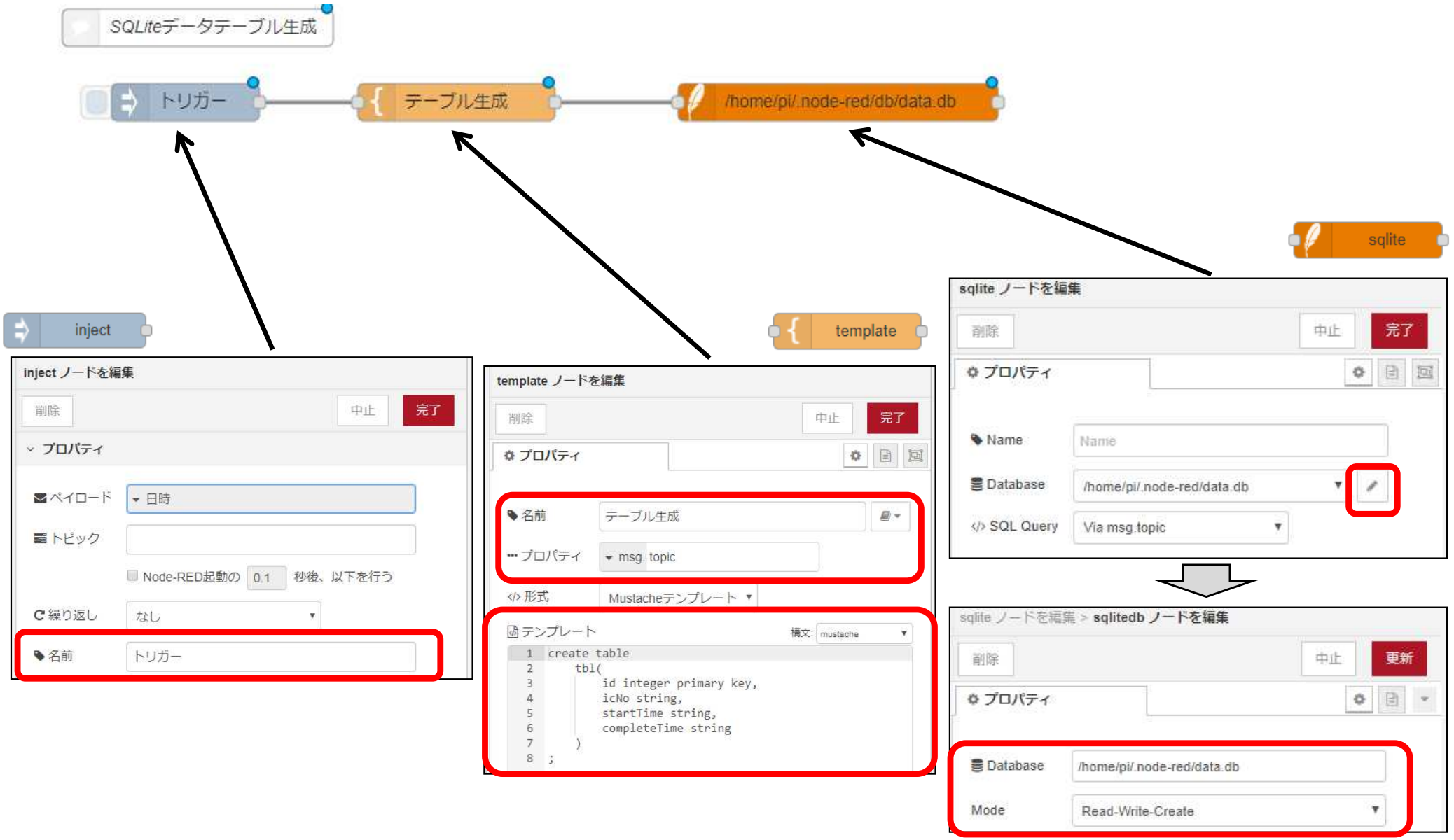
テーブル削除

```
DROP TABLE
    テーブル名;
```

データ追加／上書き

```
INSERT INTO
    テーブル名(
        カラム名1,
        カラム名2,
        ・・・・
    )
    VALUES(
        データ1,
        データ2,
        ・・・・
    );
```

テーブル生成


SQLiteデータテーブル生成
トリガー
テーブル生成
/home/pi/.node-red/db/data.db
inject
template
sqlite
inject ノードを編集
削除
中止
完了
プロパティ
ペイロード
日時
トピック
Node-RED起動の
0.1
秒後、以下を行う
繰り返し
なし
名前
トリガー
template ノードを編集
削除
中止
完了
プロパティ
名前
テーブル生成
プロパティ
msg. topic
形式
Mustacheテンプレート
テンプレート
構文:
mustache
1 create table
2 tbl(
3 id integer primary key,
4 icNo string,
5 startTime string,
6 completeTime string
7)
8 ;
sqlite ノードを編集
削除
中止
完了
プロパティ
Name
Name
Database
/home/pi/.node-red/data.db
SQL Query
Via msg.topic
sqlite ノードを編集 > sqlitedb ノードを編集
削除
中止
更新
プロパティ
Database
/home/pi/.node-red/data.db
Mode
Read-Write-Create

テーブル生成

ここ押すと起動

テーブル作成

```
create table
    tbl(
        id integer primary key,
        icNo string,
        startTime string,
        completeTime string
    );
```

テーブル名

行番号

NFCカードNo.

NFCカードを置いた時刻

NFCカードを離した時刻

フォルダ構成

home
↳pi
↳.node-red
↳data.db

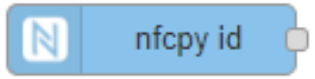
※ テーブルイメージ

id	icNo	startTime	complete Time

もくじ

NFCリーダーノードについて

NFCリーダー用のノードには、下記のような特徴があります。



- NFCカードを置いている間読み取る。
- NFCカードの読み取り内容は、IDのみ。
- 1秒ごとに読み取る。

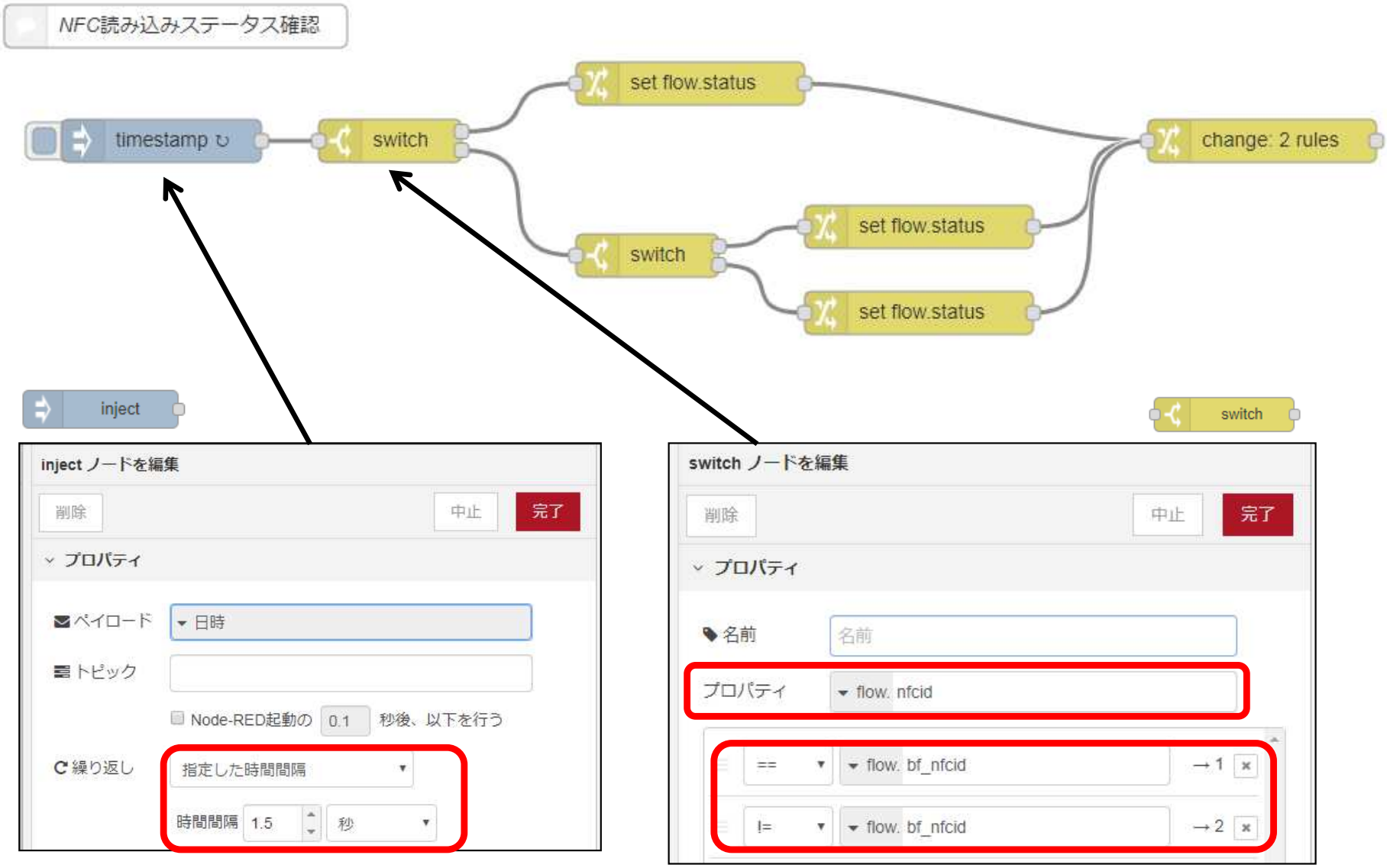
NFCカードを置いた時間と外した時間を取得して、進捗管理をしたいができない。

よって、定周期で監視し、NFCカードを置いたときのステータスと
外したときのステータスを取得する。

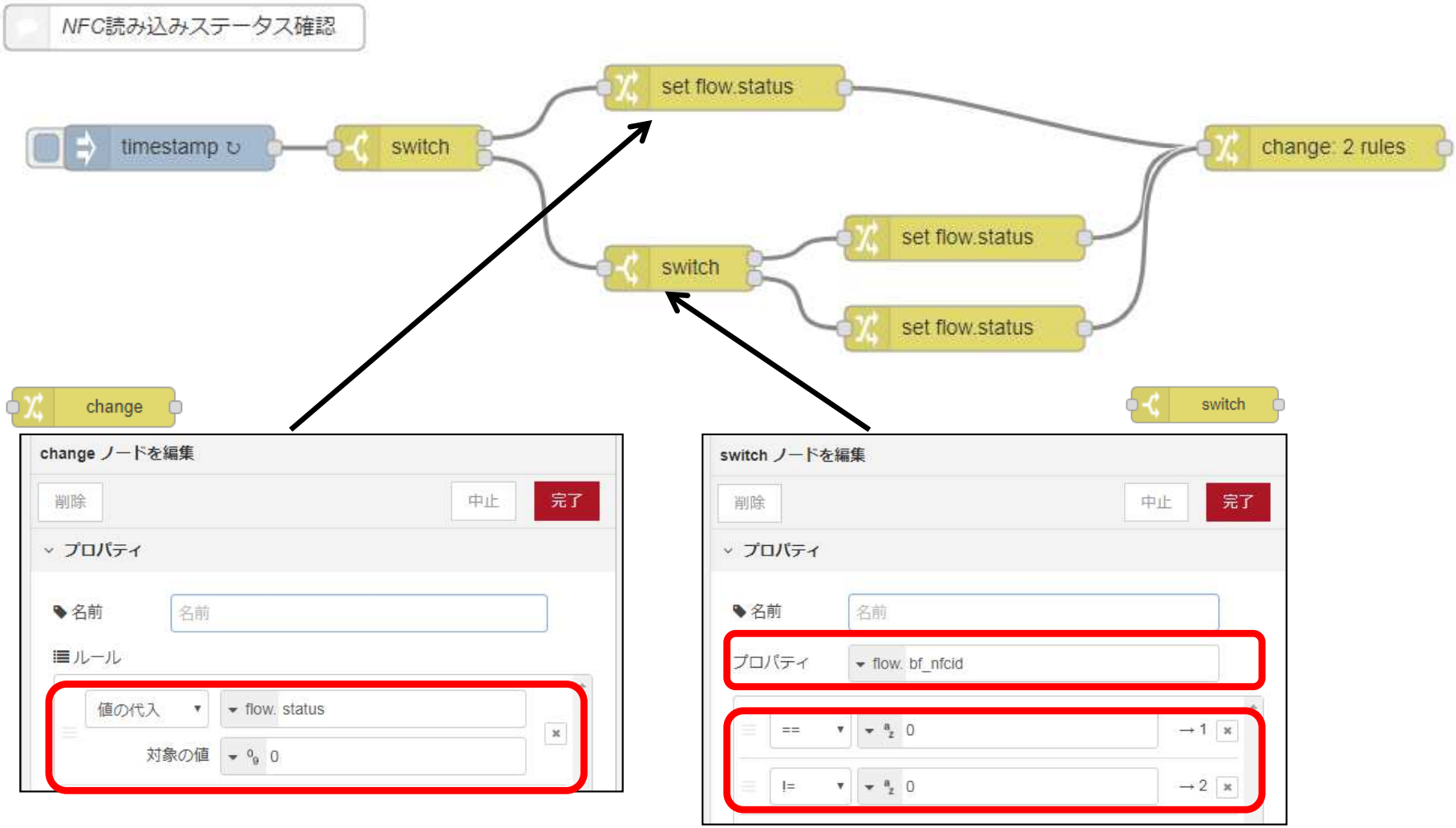
NFC読み込み


NFC読み込み
nfcpy-id
waiting
change: 2 rules
nfcpy id
nfcpy-id ノードを編集
削除
中止
完了
プロパティ
waitTime
1
Name
Name
change
change ノードを編集
削除
中止
完了
プロパティ
名前
名前
ルール
値の代入
flow. nfcid
対象の値
msg. payload
値の代入
flow. id
対象の値
msg. payload

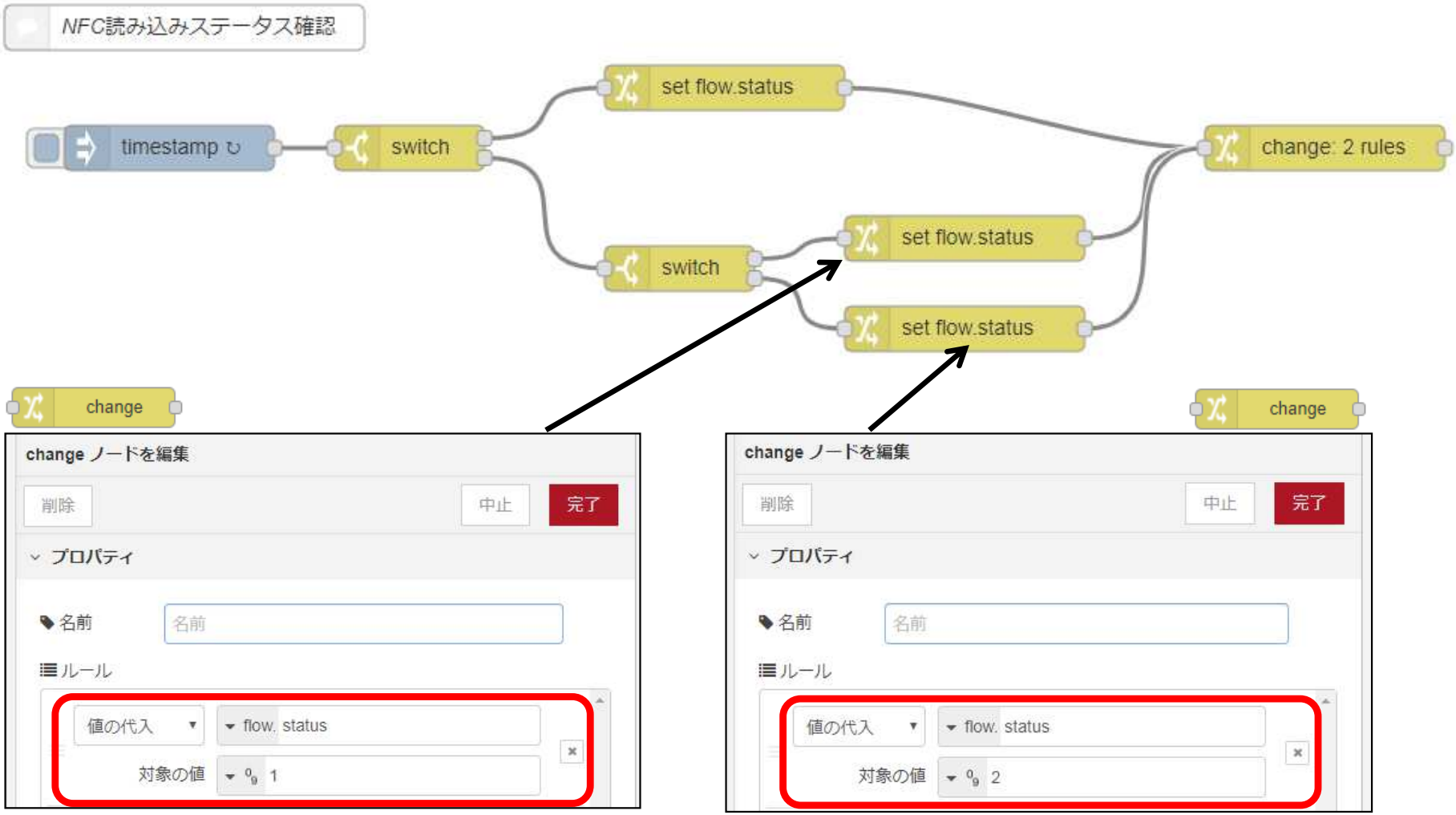
NFC読み込みステータス確認



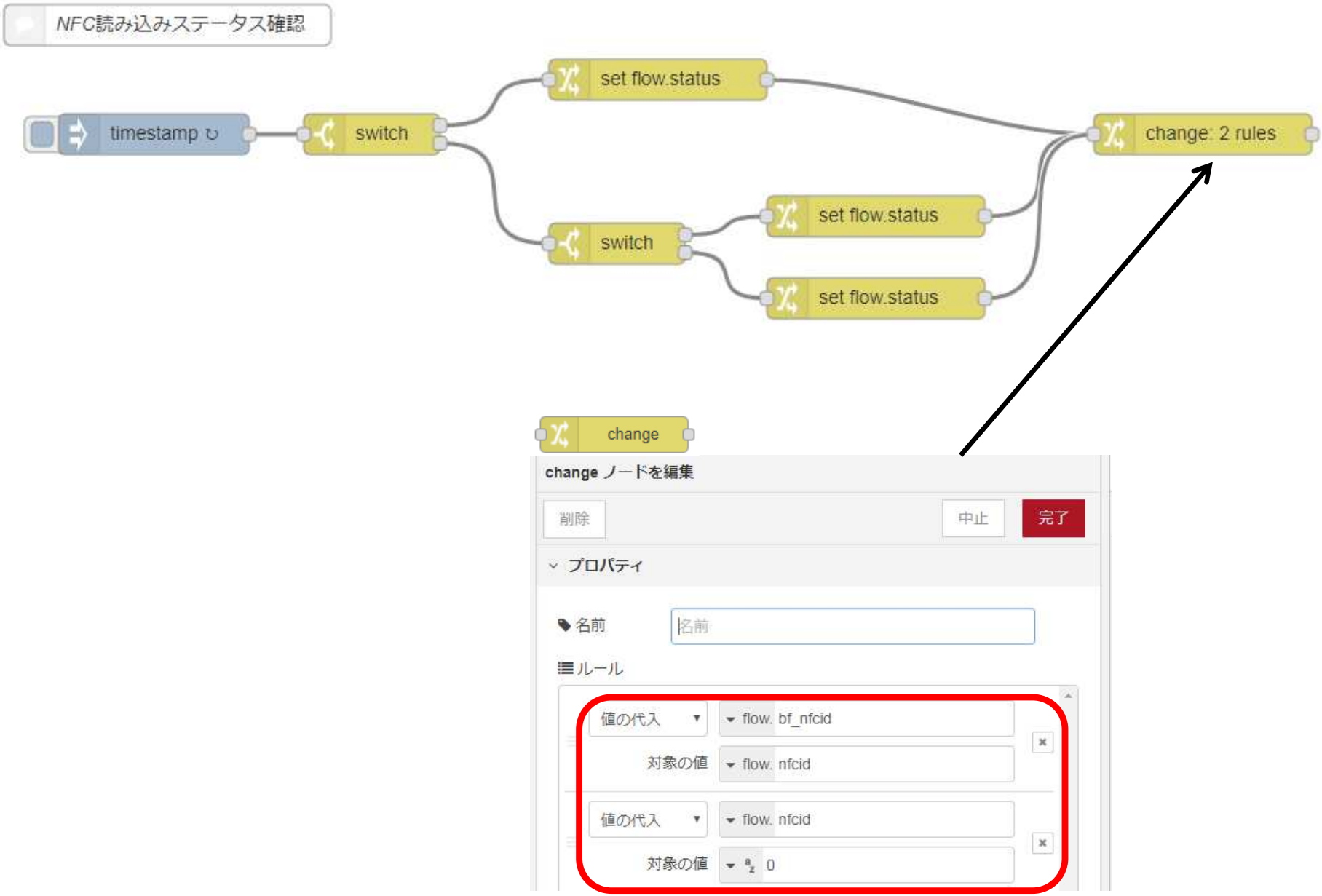
NFC読み込みステータス確認



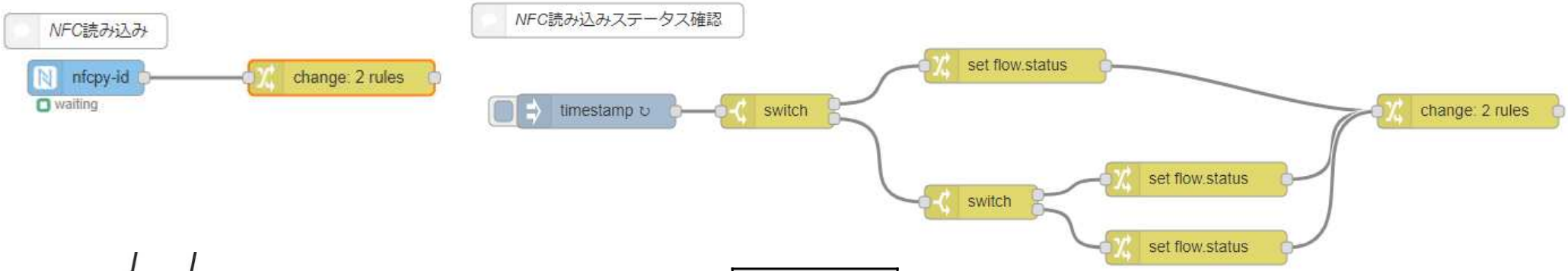
NFC読み込みステータス確認



NFC読み込みステータス確認



NFC読み込み（解説）



1秒周期

msg.
payload

flow.
nfcid

Status確認で使用

flow.
id

データベース格納時に使用

1.5秒周期

flow.		flow.
nfcid	=	bf_nfcid

前回読み込んだIDと今回読み込んだIDが同じ？

YES

flow.
status = 0

IDに変化がない

No

flow.	=	0
bf_nfcid		

IDに変化があった

？

YES

flow.
status = 1

カードを置いたとき

No

flow.
status = 2

カードを外したとき

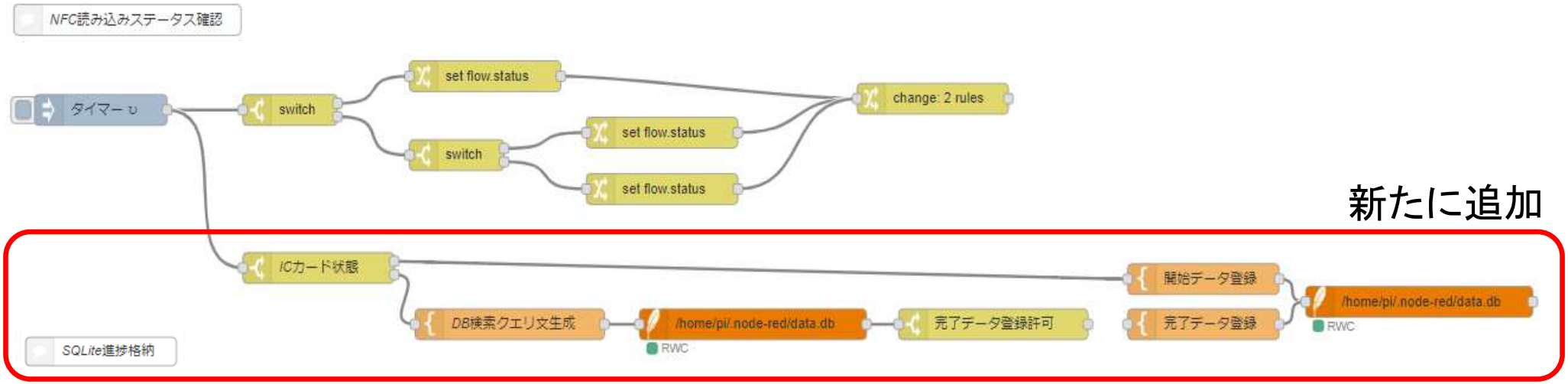
flow.
nfcid

flow.
bf_nfcid

flow.
nfcid=0

もくじ

実績収集



SQLiteのテーブルにデータを格納する操作を行います。

データ追加／上書き

```
INSERT INTO
    テーブル名(
        カラム名1,
        カラム名2,
        ・・・・
    )
    VALUES(
        データ1,
        データ2,
        ・・・・
    );
```

変数:Data1

2019/06/29

column1に追加

変数:Data2

30

column2に追加

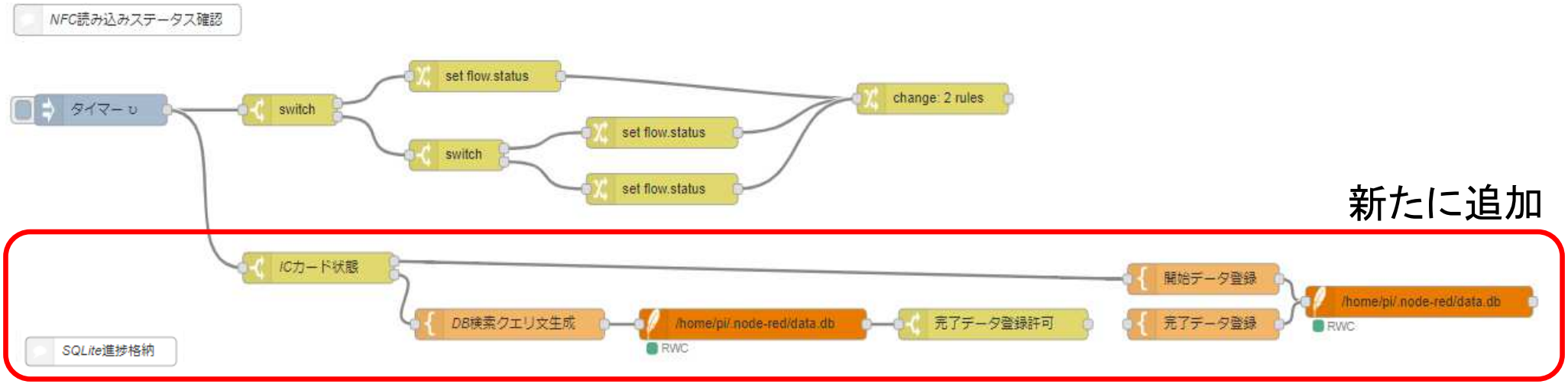
テーブル名:Table

column1	column2	column3
2019/06/29	15	200

イメージをコードにすると…

```
Insert  into
    Table(
        column1,
        column2
    )
Values(
    “{{Data1}}”,
    “{{global.currenttime}}”
);
```

実績収集



SQLiteのテーブルに格納されたデータの更新を行います。

データ更新

```
UPDATE
    テーブル名
SET
    データ
WHERE
                条件1
    and         条件2
    and         条件3
;
```

変数:Data1

30

Column1が2019/06/29の
column2の値を更新

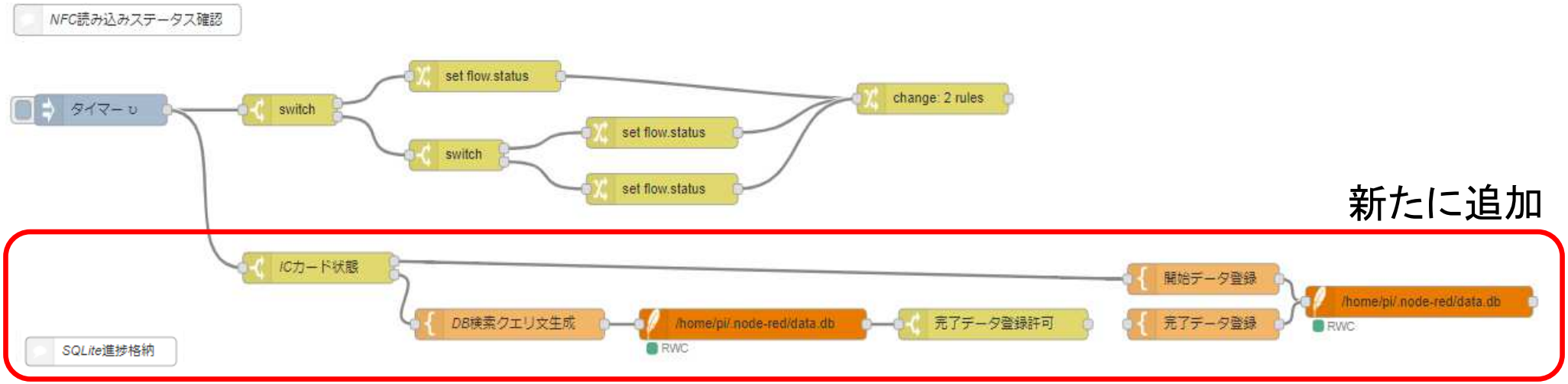
テーブル名:Table

column1	column2	column3
2019/06/29	15	200
2019/06/30	30	100
2019/07/01	20	200
2019/07/02	30	150

イメージをコードにすると…

```
update
    Table
set
    column2 = “{{Data1}}”
Where
    column1 = “2019/06/29”
;
```

実績収集



SQLiteのテーブルに格納されたデータを検索する操作を行います。

データ検索

```
SELECT
    表示カラム名
FROM
    テーブル名
WHERE
                条件1
    and         条件2
    and         条件3
;
```

テーブル名:Table

column1	column2	column3
2019/06/29	15	200
2019/06/30	30	100
2019/07/01	20	200
2019/07/02	30	150

変数:Data1

2019/07/01

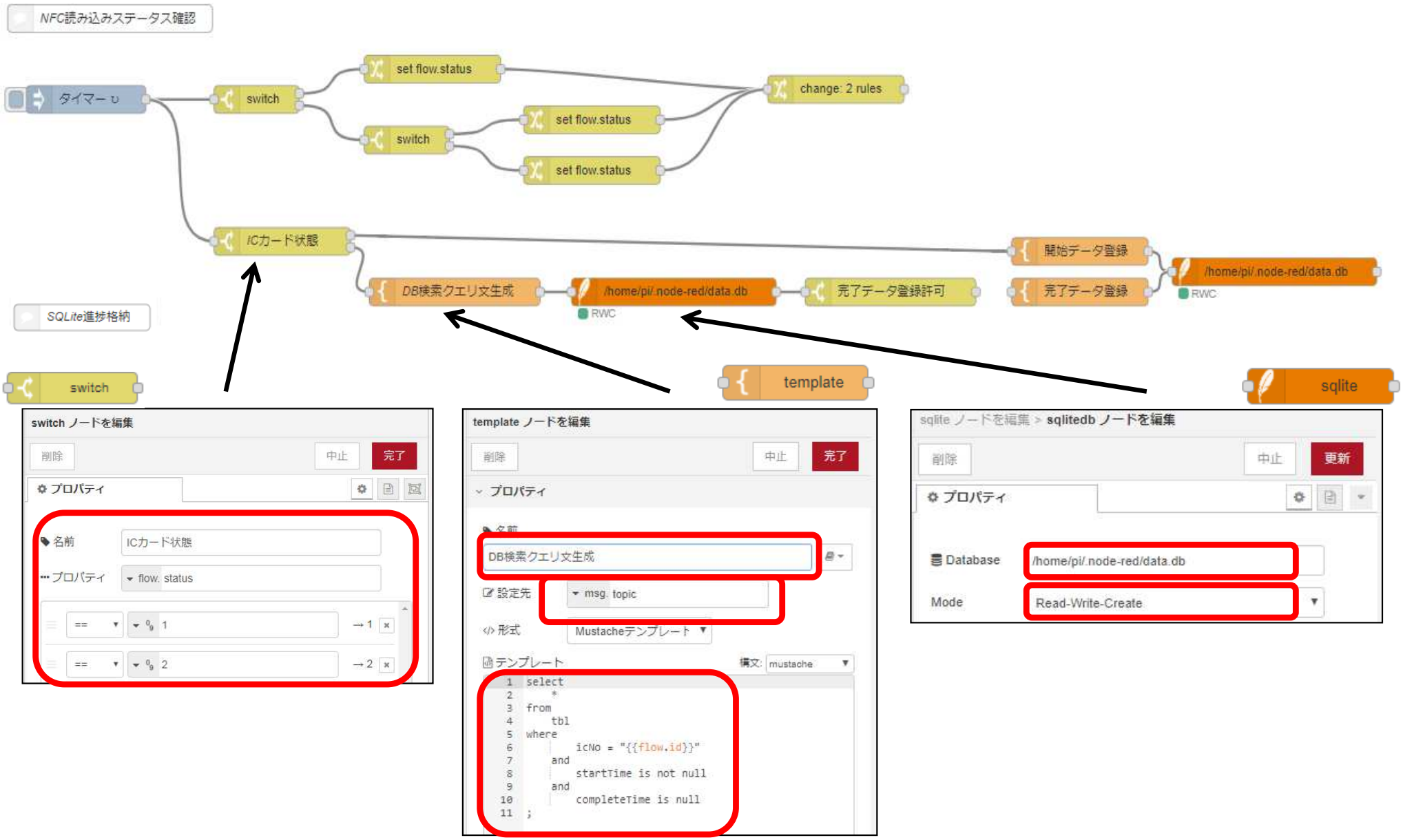
? どこにある？

2019/07/01	20	200

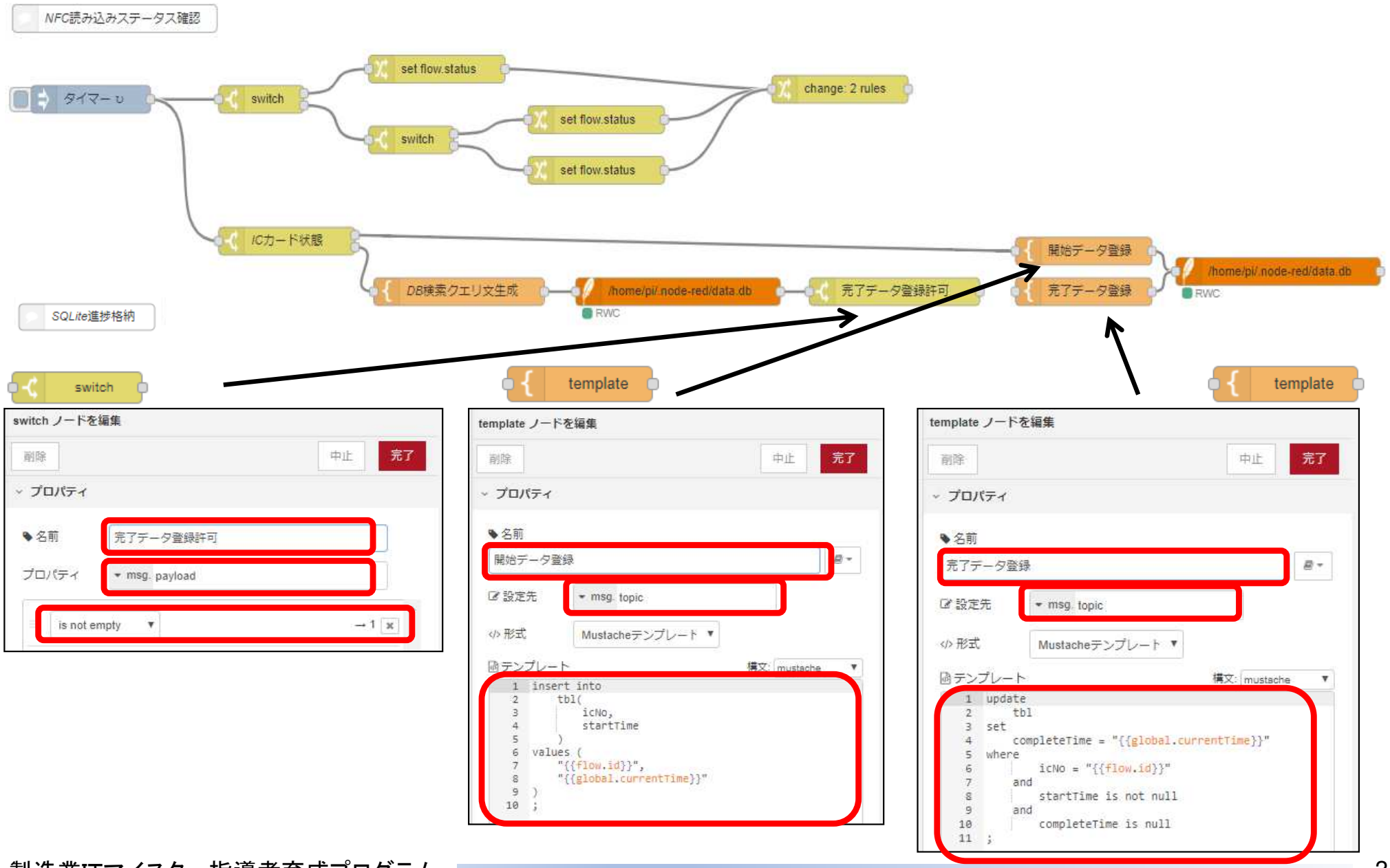
イメージをコードにすると…

```
Select
    *
From
    Table
Where
            column1 = “{{Data1}}”
;
```

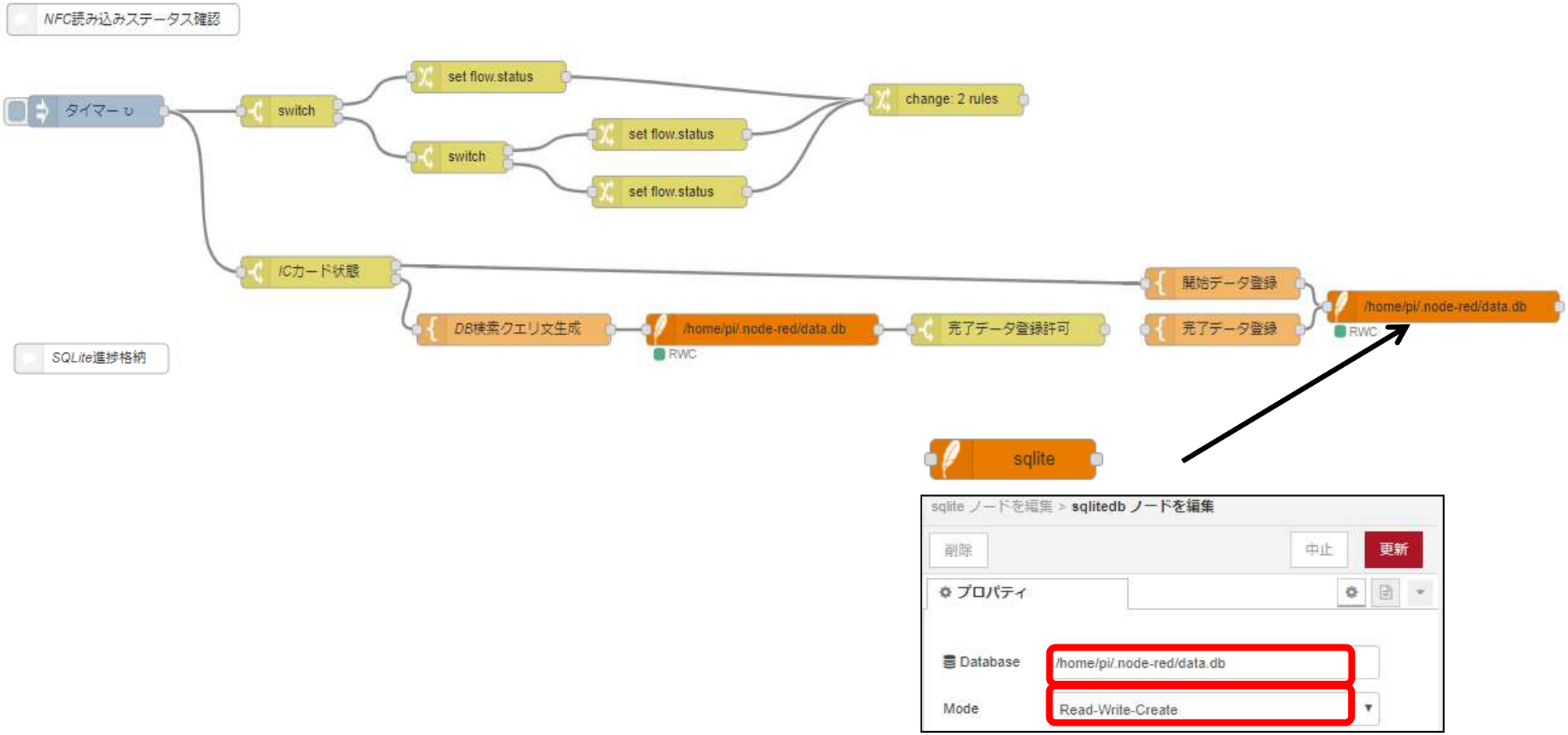
実績収集


NFC読み込みステータス確認
タイマー
switch
set flow.status
switch
set flow.status
set flow.status
change: 2 rules
ICカード状態
開始データ登録
/home/pi/.node-red/data.db
RWC
DB検索クエリ文生成
/home/pi/.node-red/data.db
RWC
完了データ登録許可
完了データ登録
SQLite進捗格納
switch
template
sqlite
switch ノードを編集
削除
中止
完了
プロパティ
名前
ICカード状態
プロパティ
flow. status
==
1
→ 1
==
2
→ 2
template ノードを編集
削除
中止
完了
プロパティ
名前
DB検索クエリ文生成
設定先
msg. topic
形式
Mustacheテンプレート
テンプレート
構文: mustache
select
*
from
tbl
where
icNo = "{{flow.id}}"
and
startTime is not null
and
completeTime is null
;
sqlite ノードを編集 > sqlitedb ノードを編集
削除
中止
更新
プロパティ
Database
/home/pi/.node-red/data.db
Mode
Read-Write-Create

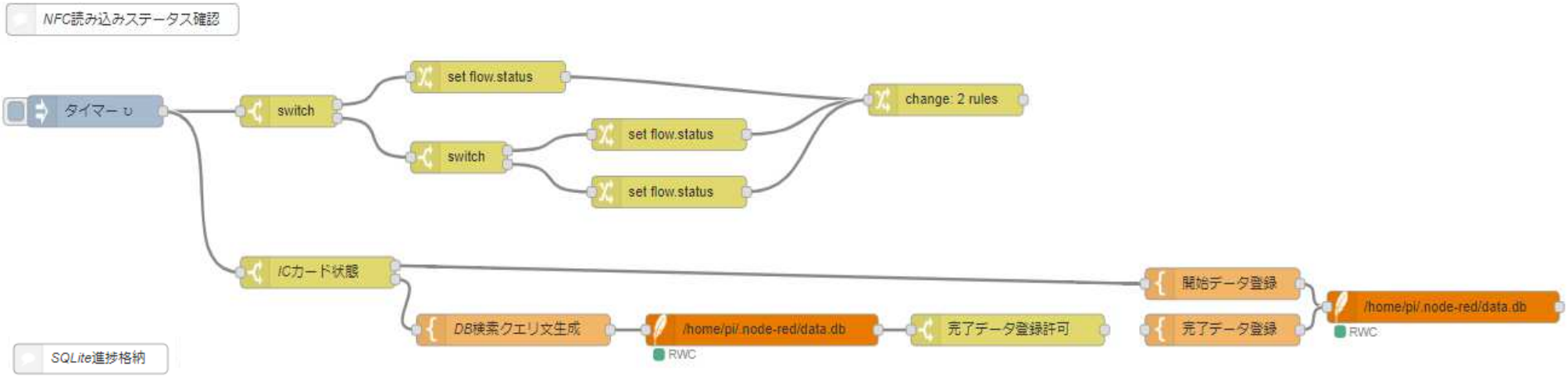
実績収集


NFC読み込みステータス確認
タイマー
switch
set flow.status
switch
set flow.status
set flow.status
change: 2 rules
ICカード状態
DB検索クエリ文生成
/home/pi/.node-red/data.db
RWC
完了データ登録許可
開始データ登録
完了データ登録
/home/pi/.node-red/data.db
RWC
SQLite進捗格納
switch
template
template
switch ノードを編集
削除
中止
完了
プロパティ
名前
完了データ登録許可
プロパティ
msg. payload
is not empty
1
template ノードを編集
削除
中止
完了
プロパティ
名前
開始データ登録
設定先
msg. topic
形式
Mustacheテンプレート
テンプレート
構文: mustache
insert into
tbl(
icNo,
startTime
)
values (
"{{flow.id}}",
"{{global.currentTime}}"
)
;
template ノードを編集
削除
中止
完了
プロパティ
名前
完了データ登録
設定先
msg. topic
形式
Mustacheテンプレート
テンプレート
構文: mustache
update
tbl
set
completeTime = "{{global.currentTime}}"
where
icNo = "{{flow.id}}"
and
startTime is not null
and
completeTime is null
;

実績収集


NFC読み込みステータス確認
タイマー 0
switch
set flow.status
switch
set flow.status
set flow.status
change: 2 rules
ICカード状態
開始データ登録
/home/pi/.node-red/data.db
RWC
DB検索クエリ文生成
/home/pi/.node-red/data.db
RWC
完了データ登録許可
完了データ登録
SQLite進捗格納
sqlite
sqlite ノードを編集 > sqlitedb ノードを編集
削除
中止
更新
プロパティ
Database
/home/pi/.node-red/data.db
Mode
Read-Write-Create

実績収集



global.
currenttime

NFCカードNo.と開始時刻を追記

flow.
ld

置いたとき

flow.
status

テーブル名：tbl

id	icNo	startTime	complete Time

NFCカードNo.を検索

離したとき

flow.
id

global.
currenttime

見つかった行に完了時刻を追記

もくじ

ダッシュボードでリストを表示する方法

DBで取得したデータを表に表示

作成方法：**HTMLコード**に下記を記載します。

```
表タイトル
<table style="width:100%" Border="solid">
  <tr>
    <th>項目名１</th>
    <th>項目名２ </th>
    <th>項目名３</th>
  </tr>
  <tr ng-repeat="x in msg.payload">
    <td>{{msg.payload[$index].item1}}</td>
    <td>{{msg.payload[$index]. item2}}</td>
    <td>{{msg.payload[$index]. item3}}</td>
  </tr>
</table>
```

テーブル（表）
<table> ～ </table>

テーブルのスタイル
style=***

画面に対するテーブルの幅
width:”***%”

線の種類を指定
Border=“***”
※solid：1本線で表示

実績一覧

工程No	作業	開始時間	完了時間
1	A	2019-01-21 18:31:12	2019-01-21 18:31:17
1	A	2019-01-21 17:28:20	2019-01-21 17:28:27
1	A	2019-01-21 17:28:09	2019-01-21 17:28:14
1	A	2019-01-21 16:53:21	2019-01-21 16:53:26

ダッシュボードでリストを表示する方法

DBで取得したデータを表に表示

作成方法：**HTMLコード**に下記を記載します。

```
表タイトル
<table style="width:100%" Border="solid">
  <tr>
    <th>項目名１</th>
    <th>項目名２ </th>
    <th>項目名３</th>
  </tr>
  <tr ng-repeat="x in msg.payload">
    <td>{{msg.payload[$index].item1}}</td>
    <td>{{msg.payload[$index]. item2}}</td>
    <td>{{msg.payload[$index]. item3}}</td>
  </tr>
</table>
```

1行目:列の設定
<tr> ～ </tr>

テーブルヘッダー
<th> ～ </th>

繰り返し行う
ng-repeat=“ A in B”
A：仮変数
B：繰り返し対象の配列

実績一覧

工程No	作業	開始時間	完了時間
1	A	2019-01-21 18:31:12	2019-01-21 18:31:17
1	A	2019-01-21 17:28:20	2019-01-21 17:28:27
1	A	2019-01-21 17:28:09	2019-01-21 17:28:14
1	A	2019-01-21 16:53:21	2019-01-21 16:53:26

ダッシュボードでリストを表示する方法

DBで取得したデータを表に表示

作成方法：**HTMLコード**に下記を記載します。

```
表タイトル
<table style="width:100%" Border="solid">
  <tr>
    <th>項目名1</th>
    <th>項目名2 </th>
    <th>項目名3</th>
  </tr>
  <tr ng-repeat="x in msg.payload">
    <td>{{msg.payload[$index].item1}}</td>
    <td>{{msg.payload[$index]. item2}}</td>
    <td>{{msg.payload[$index]. item3}}</td>
  </tr>
</table>
```

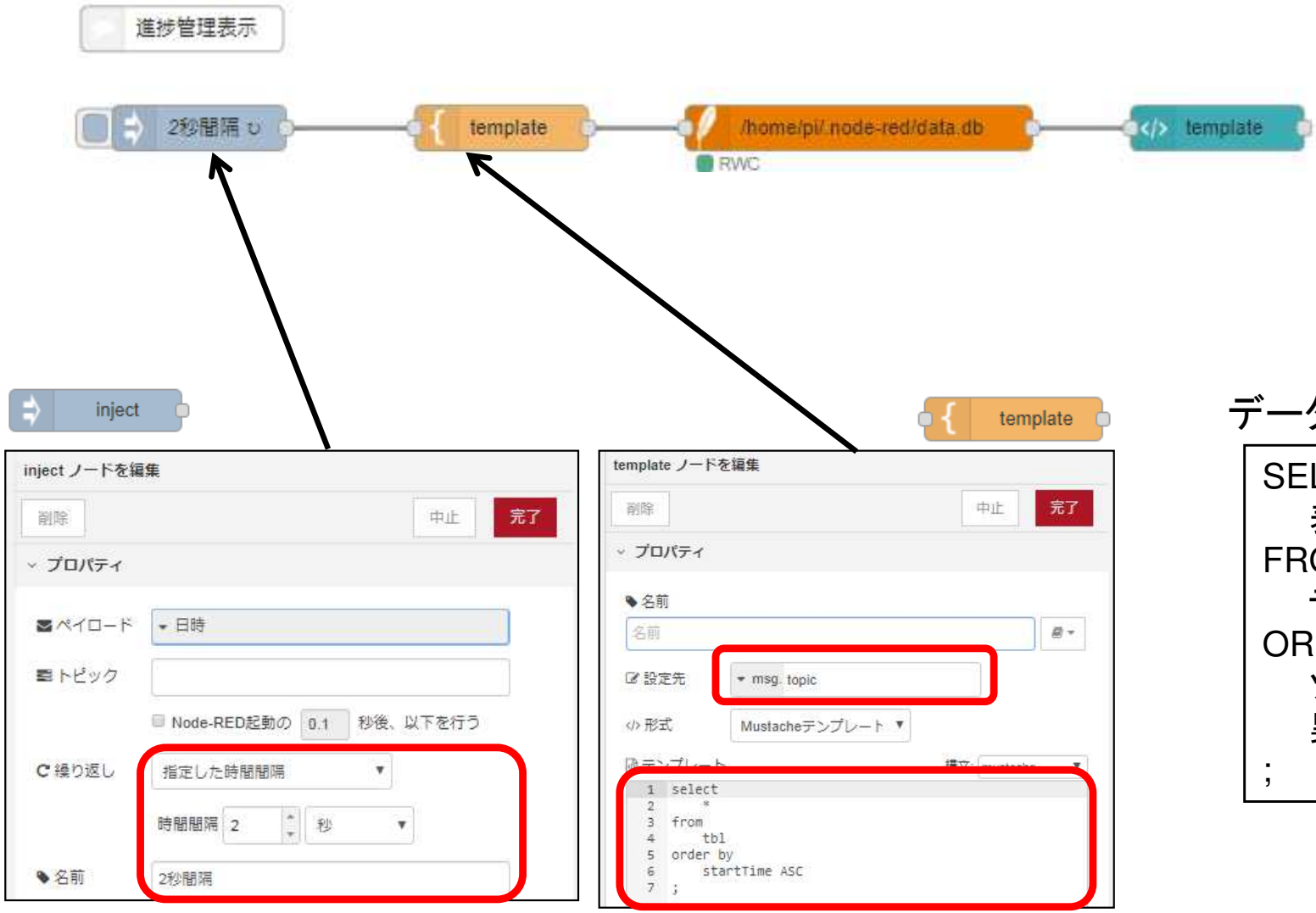
テーブルデータ
<td> ～ </td>
$index：繰り返しの時に
1ずつ増えていく
例
1回目　$index=1
2回目　$index=2　・・・

item*：変数名

実績一覧

工程No	作業	開始時間	完了時間
1	A	2019-01-21 18:31:12	2019-01-21 18:31:17
1	A	2019-01-21 17:28:20	2019-01-21 17:28:27
1	A	2019-01-21 17:28:09	2019-01-21 17:28:14
1	A	2019-01-21 16:53:21	2019-01-21 16:53:26

ダッシュボード作成

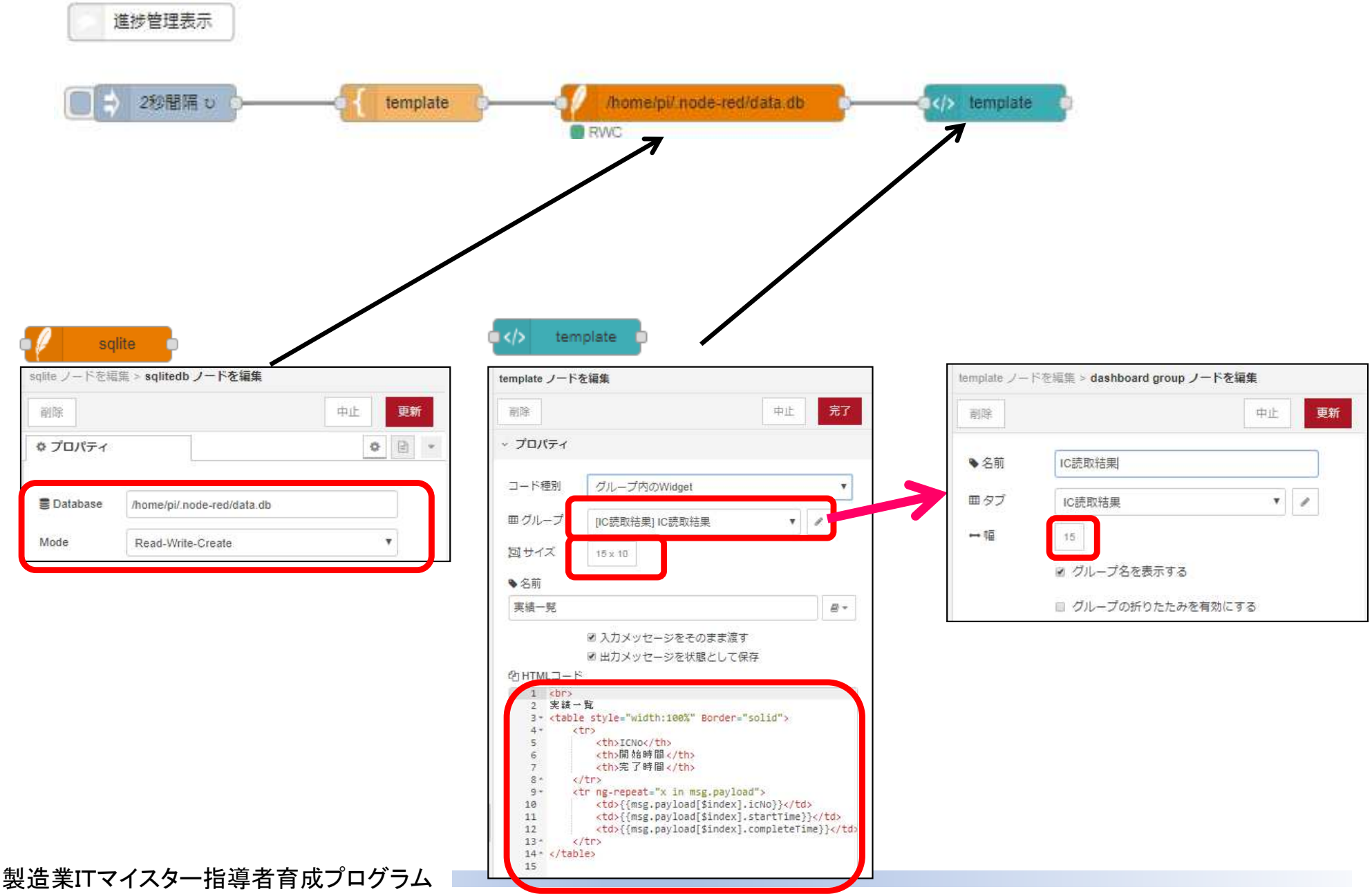


データソート

```
SELECT
    表示カラム名
FROM
    テーブル名
ORDER  BY
    ソートするカラム名
    昇順/降順
;
```

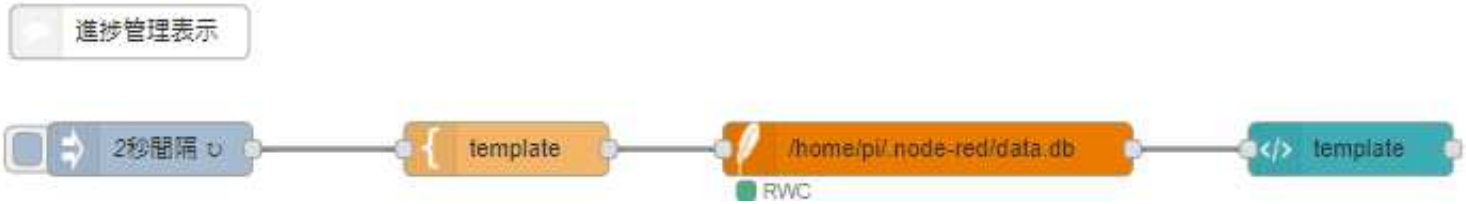
開始時間順にソートして、リストを見やすくします。
昇順：ASC、 降順：DESC

ダッシュボード作成


進捗管理表示
2秒間隔
template
/home/pi/.node-red/data.db
RWC
template
sqlite
sqlite ノードを編集 > sqlitedb ノードを編集
削除
中止
更新
プロパティ
Database
/home/pi/.node-red/data.db
Mode
Read-Write-Create
template
template ノードを編集
削除
中止
完了
プロパティ
コード種別
グループ内のWidget
グループ
[IC読取結果] IC読取結果
サイズ
15 x 10
名前
実績一覧
入力メッセージをそのまま渡す
出力メッセージを状態として保存
HTMLコード

実績一覧
<table style="width:100%" Border="solid">
<tr>
<th>ICNo</th>
<th>開始時間</th>
<th>完了時間</th>
</tr>
<tr ng-repeat="x in msg.payload">
<td>{{msg.payload[$index].icNo}}</td>
<td>{{msg.payload[$index].startTime}}</td>
<td>{{msg.payload[$index].completeTime}}</td>
</tr>
</table>
template ノードを編集 > dashboard group ノードを編集
削除
中止
更新
名前
IC読取結果
タブ
IC読取結果
幅
15
グループ名を表示する
グループの折りたたみを有効にする

ダッシュボード作成（解説）



テーブル名：tbl

id	icNo	startTime	complete Time

① startTimeの列を昇順にソートする

② 取り出す

Msg.payload			
id	icNo	startTime	Complete Time

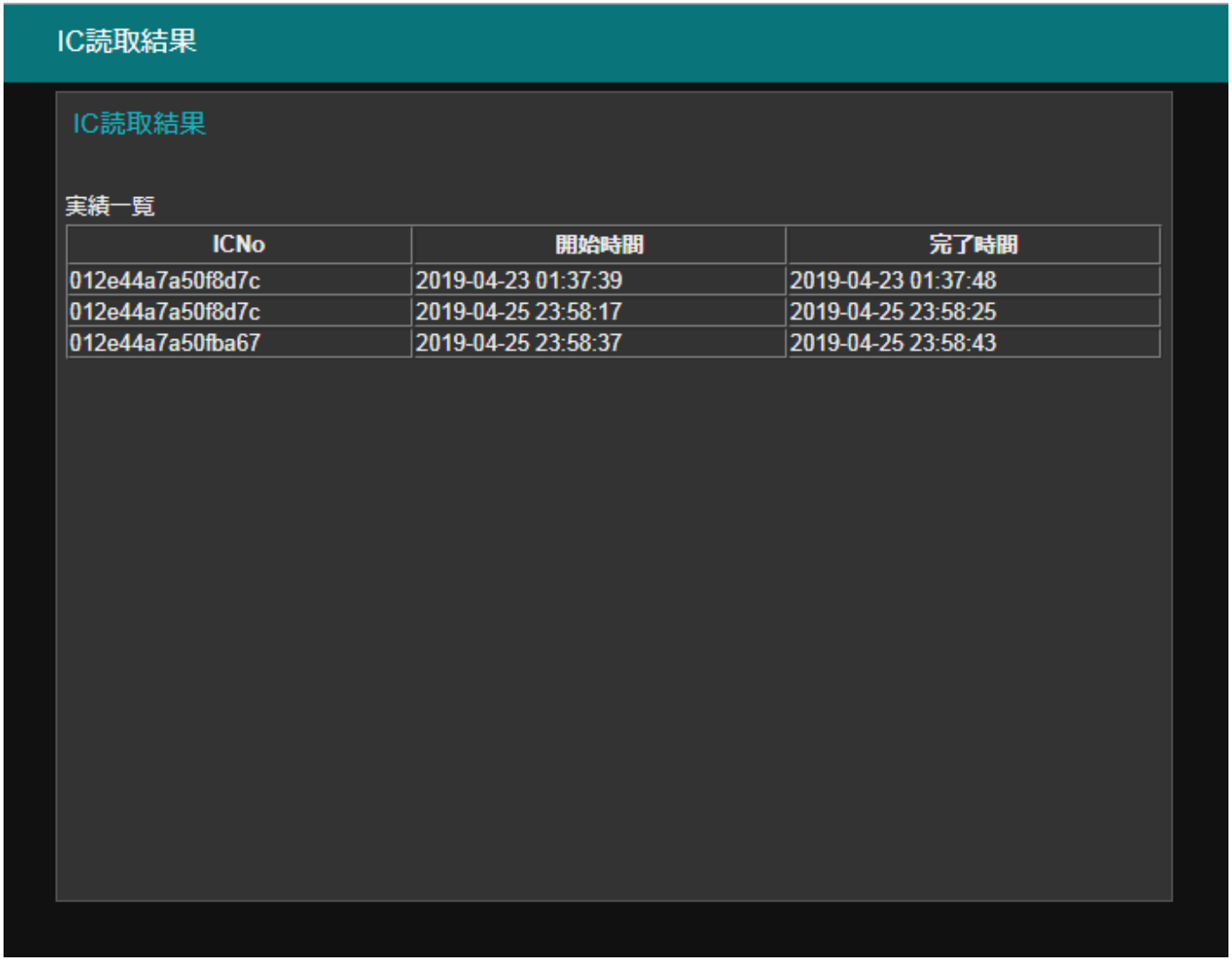
③ ダッシュボードに表示

icNo	開始時間	終了時間

見える化

ダッシュボードのURL：http://***.***.***.***:1880/ui

IPアドレス



本教材利用上の注意事項