SPring-8/SACLA 加速器運転パラメータ管理データベースの構築

A DATABASE SCHEME TO MANAGE OPERATIONAL POINTS AND CALIBRATION VALUES AT SPRING-8/SACLA

岡田謙介#, A), 福井達 B), 前坂比呂和 B), 田尻 泰之 C), 住友 博史 C),
Kensuke Okada #, A), Toru Fukui A), Hirokazu Maesaka B), Yasuyuki Tajiri, Hiroshi Sumitomo
A) JASRI
B) RIKEN
C) SES

Abstract

Taking the opportunity that the control framework of SPring-8/SACLA accelerators is replaced toward upgrade projects, we developed a new database scheme to manage operational points and calibration values. In the scheme, each parameter group has a work area and a storage area in the relational database (RDB). At the end of each tuning step, the data in the work area are copied to the storage area with a proper label at any point of their tree structure for later use. The RDB tables are designed to handle a general format, so that the common access functions are available. One of challenges in the deployment is to make many user-level applications reusable. As an example, the calibration values of the beam position monitors were used to be scattered around in tables of various formats. Those are now rearranged in the same structure by assigning each aspect to an identifier like eqip_id, element_id, and bpmckt_id, so that the manageability has been greatly improved. Since 2018, the new scheme has been used in the operation.

1. はじめに

加速器運用において、電磁石やモニターなどの設定値、校正値を管理し、後日加速器状態の再現や、比較のために見返す必要がある。本稿では運転パラメータの管理と呼ぶことにする。

SPring-8/SACLA では機器の制御に、早い段階からリレーショナルデータベース(RDB)を組み込んでおり[1]、運転パラメータの管理にデータベースを利用してきた。ところが、場当たり的な対応が積み重なって、特に運用の歴史の長い SPring-8 では、テキストファイルで管理している場合や、RDB を利用していても、項目毎に違う種類のテーブルを作成し、アクセス関数をその都度提供したことで非常に見通しの悪い状態になっていた。

近年ストレージリングのアップグレード計画が立ち上がり[2]、SACLA の高性能 linac を SPring-8 の入射器として利用する準備が進んでいる[3]。加速器の制御系においても、これまでのシステムの入れ替えを進めてきた[4]。運転パラメータの管理についても、この機会に見通しの良い形への整理を計画し、定常運転にたどりついた。

本稿は、この運転パラメータの管理について、まずこれまでの利用状況の分析から要件を洗い出し、立てた方針を説明する(2 節)。RDB のテーブル設計(3 節)と運転 GUI で利用するアクセス関数の設計(4 節)に続いて、制御システム移植の段取りといくつかの特殊例について(5 節)述べる。今後の展望(6 節)と課題(7 節)にふれて、まとめ(8 節)とする。

k.okada@spring8.or.jp

2. 要件の洗い出しと方針

2.1 管理の単位

SPring-8/SACLA 制御方針では、制御系上位からの命令は機器毎に独自の名称を通じてやり取りされる。従って、基本的には運転パラメータは機器の識別子(eqip_id)に紐づけた属性で管理できる。しかし、それだけでは使い勝手が悪いことがある。例えば、複数の電磁石が一つの電源につながっている場合、電源の eqip_id の電流値という機器側の見方と、軌道計算のための加速器モデル側の見方があり、両者の接続には、加速器の構成要素(セル xx の Q1など)と例えばエネルギーで規格化した電磁石の強さと電流値の変換係数、束縛条件としてシリーズの組み方が必要である。似た例として、Beam position monitor (BPM)について、機器側は複数チャンネルを束ねて扱う読み出し回路毎の設定値、校正値に落とし込まれるが、Closed orbit distortion (cod)補正のフィードバックの計算や、一時的なマスクの操作は加速器構成要素側からのアプローチになる。

これまでの扱いでは多種の管理単位が入り乱れている状態だった。今回、機器の識別子(eqip_id)以外の識別子を使う場合の利用方法を整理し、また識別子どうしの関係性も一般化した形で保存することとした。これまでセルと連番の組み合わせ等で複数インデックスを使用している例があったが、複雑になることを避けるため、一つにまとめることを基本方針とした。

2.2 現運転パラメータと履歴

加速器の運転サイクル変更の際や試験時に、迅速に過去の運転パラメータを適用して、調整の出発点とすることが要求される。そのためには、あらゆる変更を逐一記録する必要はなく、一連の調整の区切り毎にラベルを付

けて保存できれば良い。従って秒以下の更新頻度は想定しない。

作業用のエリアと、日時と紐づけて保存するエリアを用意し、両者の出し入れを行う。

調整によっては、一部の運転パラメータのみ別の期間の設定を読み込むといった需要がある。グループ（例えばステアリング磁石）毎の扱い、上のグループ（例えば加速器全体）で一括の扱いが必要である。SACLA の振り分け運転では、ビームライン毎の調整ノブが存在する。そのためにグループの階層構造を取り入れる。

2.3 懸念事項

これまでの RDB のテーブル構造は、必要に応じて用途ごとに専用の設計で、読み出しについてもストアドプロシージャーを間に挟んで専用の読み出し関数を作成してきた。今回一般的な枠組みで取り扱うにあたって、管理テーブルを参照する必要が出てくるので、同じ条件下ではどうしてもアクセス速度が遅くなると予想される。

また、多次元のインデックスで管理するテーブルを単純な型に展開できるのか、先の可能性を含めた完全な見通しが立てられないまま進める必要があった。

3. RDB テーブル設計

3.1 Entity and Value (EAV) スタイルのデータ管理

識別子の色々な属性に紐づけて、運転パラメータの値を保存するために、データを EAV スタイルで構築することにした。これは一般に RDB のテーブル設計では、ふさわしくないとされる[5]。しかし、今回の利用側の扱いは作業エリアの全体、もしくは履歴のバージョン番号で指定されたデータセットをまとめて利用するのが基本で、各行の情報は独立しており集計の必要もないことから、次節に述べるデータベースアクセス関数を統一できる利点が勝ると判断した。調整グループ毎に作業エリアと保存エリアのテーブルを作成する(Fig. 1).

Figure 1: (Left) Data table contents of working area. “change_flg” and “update_flg” are used when moving data from/to the stored area. (Right) Data table contents of stored area. Another table connects version_id with “comment” and set/run time.

3.2 識別子と属性

各データは識別子（prim_id）と属性（pkey_id）に紐づけて値を保持する。新規グループ作成時にどの識別子で整理するかを決定し、グループ管理テーブルに記述する。属性管理テーブルは、識別子毎に使用する属性を定義する。例えばマグネットの最大電流値を保存しておく場合は識別子 eqip_id = ’sr_mag_ps_b の整理番号’と属性’current_max’のペアにに実数値を結びつける。

識別子として利用するためには、名称との対応関係（例、eqip_id と eqip_name）を収めたテーブルを指定することとし、加えて 2 つの識別子どうしを結ぶための ID-ID 管理テーブルの規定を行った。ここでは多対多の接続を定義する。

3.3 階層構造の構築

部分的な運転パラメータの取り扱いを実現するためにグループに階層構造を持たせた。上位の階層からは、紐づけた下位のグループを一括で取り扱うことができる。グループ間の階層構造は Tree 閉包テーブルに記述した（Fig. 2 左）。親子の関係には 2 種類用意し、「継承」の関係の場合は、グループは分かれているが読み出し時は一つのグループのように扱う。

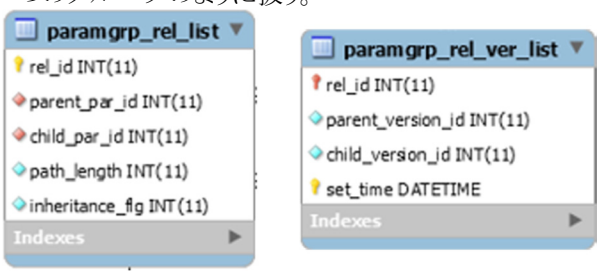

Figure 2: (Left) Management of tree structure. Relations are stored by a closure table. “inheritance_flg” controls the inheritance feature which treats two groups coupled. (Right) Table structure to store versions.

3.4 バージョン管理

更新の際は、階層構造で下位グループの更新を伴うが、各グループの保存頻度が異なるため、ある時点での最新バージョン番号は異なる。上位グループからの保存時は Tree 閉包テーブルの親子関係それぞれについてバージョン番号を記録した（Fig. 2 右）。

3.5 運転パラメータグループの削除

設計の見直しなどの理由で、一度登録した運転パラメータグループを取り消したい場合が出てくる。しかし管理テーブルでは登録順にグループに番号を振り、それを元に階層構造を構築しているため、そのあとに登録したグループに影響なくなかったことにするのは難しい。そこで、コロン（“:”）に特別な意味を持たせ、名前がコロンで始まるグループはアクセス関数で処理されない造りとし、実用上の削除扱いとした。

4. アクセス関数

加速器運転用 GUI で利用するためのデータベースアクセスライブラリを整備した。データの授受のために作業エリアのコンテンツに加え、識別子、属性の名称と型を含んだ C の構造体を用意した。

また python でのインターフェースは、コンテンツを web で確認するためと新規グループ登録作業に利用している。

4.1 読みだし

作業エリアや保存エリアにあるデータを読み出す。「継承」関係にあるグループがある場合はグループを合わせて取り扱う。

4.2 書き込み

構造体を用意し、グループ名を指定して作業エリアに書き込む。「継承」関係にあるグループ間については、明示的に指定する必要がある。前項で読み出したデータをそのまま更新した場合は正しく処理される。

4.3 保存エリアから作業エリアへ

現在の作業エリアのデータの扱いによって、3 つのモードを用意した。0: クリーンアップ後にコピー、1: 作業エリアに追加、2: 作業エリアに存在する識別子についてのみ上書き。

0 は運転サイクルの変更時やスタディの新規開始時を想定。1 は新しい機器などを追加した際に過去のデータセットと合算する場合。2 は逆に撤去した機器の設定を作業エリアにおいてしまって、命令が通らないことによる混乱を避けるものである。

4.4 作業エリアから保存エリアへ

適当な調整の区切りで保存エリアにコピーする。階層の上位からまとめてコピーする場合、データに変更のないグループについては、保存エリアへのコピーは行わず、バージョン番号を引き継ぐこととした。無駄な保存領域の増大を防ぐためで、ディスクのスナップショット機能[6]にヒントを得た。ここで、3 階層以上の場合、中間のグループのコンテンツが更新されていなくても、下のグループの更新があれば、中間のグループのバージョンをインクリメントすることになるので、更新の有無の判断には再帰的な処理が必要である。このための下処理はデータベースのストアドプロシージャーに負わせた。

4.5 ユーティリティ関数

識別子間の関係(ID-ID 管理テーブル)を参照するための関数と、作業エリアのデータに頻繁にアクセスする際に利用できる関数を用意した。どちらもキャッシュに保存して 2 回目以降はキャッシュからデータを読み込む。データ更新の際の再読み込みは、ユーザー側で行う必要があるが、これまでのアプリケーションとの移植の際に有用だった。

4.6 python スクリプトインターフェース

新規グループを追加する際は、新規テーブルの作成からグループ管理テーブルへの追加、階層管理テーブルへの追加と、一連の作業が必要である。python で SQL をラップした関数を作成して使用している。またコンテンツを web 表示するためにも利用している(Fig. 3)。スクリプト操作は管理者向けとして、誤って RDB の整合性が崩れないよう配慮している。

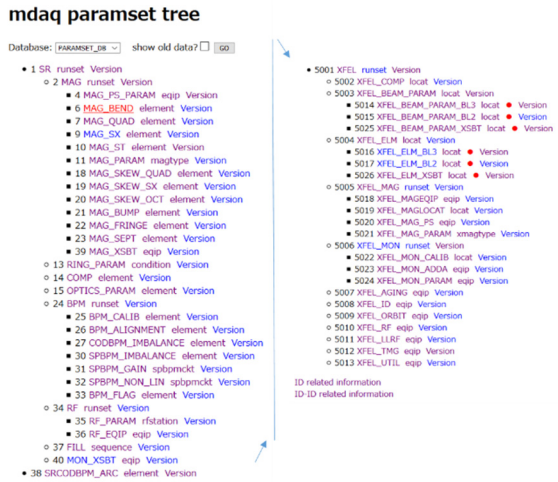

Figure 3: The front page of the web browser showing the current tree structure of parameter groups. Each is linked to data and version browser.

5. 制御系移行状況

2017 年、2018 年のメンテナンス期間を使って BL1 専用加速器、SPring-8/SACLA 加速器の制御系移行を行った。以下で、その中でいくつかの項目について述べる。

5.1 旧アプリケーションとの接続

運転パラメータ管理の移行における課題は、これまで運用に用いてきたアプリケーション資産の生かし方であった。テキストファイル管理だったものも含めデータを移動させるとともに、50 本強の wrapper 関数を作成してもとのアプリケーションとの接続を行った。

5.2 cod 定期収集データと加工データの分離

SPring-8 の約 300 個の BPM で収集される cod データは、周期的に保存するデータをもとに、加工したり、電子軌道に摂動を与えたタイミングのデータにラベル付けをしたりして、後の解析に利用している。これまではこの目的のために、加工データを周期的データと同じフォーマットで同じログデータベースに保存する関数と、読み出し関数を用意してアプリケーションで使っていた(Fig. 4 top)。今回、制御系の更新により、同じ方式は実現しづらくなったため、頻度がさほど高くない加工データについては、運転パラメータの枠組みで保存することにした(Fig. 4 bottom)。アクセス関数側で運転パラメータの探索、必要ならログデータベースへの参照を行い、上位アプリケーションの入出力は変わらないように処理を隠蔽した。

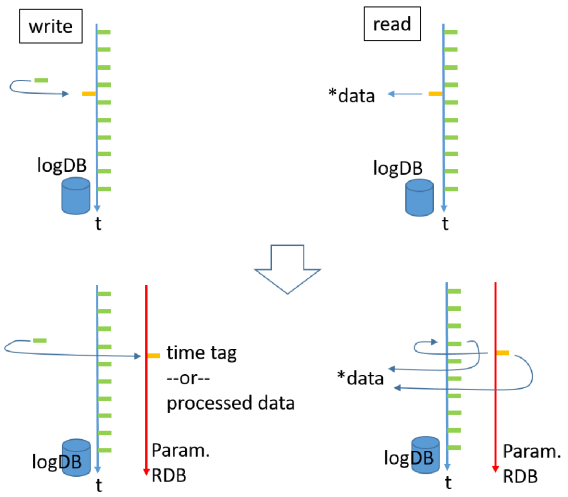

Figure 4: Modification of the cod data labeling. (Top) It used to insert processed data into the same log database for the cyclic data collection. (Bottom) Processed data or just a simple time tag is stored in the parameter database. But the interface of the access function is adjusted to be unchanged.

5.3 新しい識別子の導入

いくつかの運転パラメータについては、新たに識別子(ID)を導入して、整理を行った(Table 1)。多次元マップの運転パラメータも 1 次元に落とした。必要に応じて ID 間の対応関係を通じて変換する。Fill pattern で作成した sequence_id は単に連番でそれ自体に意味はない。今回のような順序が付いた項目や、内挿操作を伴う補正点列の保存に使用する予定である。

Table 1: List of New ID's

BPM parameters (alignment, calibration)	
Before	2D map (#cell, #bpm)
new ID	element_id (SP8), locate_id (SACLA)
SPBPM parameters (gain, non-linearity correction)	
Before	2D map (#cell, #gain_no)
new ID	spbpmckt_id
Magnet parameters (strength – current coeff.)	
Before	Text file / hard coded
new ID	magtype_id
Fill pattern	
Before	2D map (bkt_order, bkt_id)
new ID	sequence_id

5.4 アクセス速度の問題

旧アプリケーションの中で、多重ループの中でデータベーステーブルにアクセスするコーディングになっているものがあり、体感速度の大幅な低下が問題になった。キャッシュを使うユーティリティ関数を使用することで、アプリケーションロジックを大幅に作り変えることなく対応した。

6. 今後の展望

今回の制御系移行で RDB のプラットフォームを SAP-ASE(旧 Sybase)[7]から、MariaDB[8]に変更した。これにより、機器側(現在多くは VME)の Solaris OS からの DB アクセスライブラリの利用が容易になった。今回の運転パラメータの管理方法は、リアルタイム性を必要とする用途には不適だが、機器の初期設定に利用できる。上位アプリケーションで作業エリアの設定値を更新し、機器側に初期化命令を発行することで、設定の記録を残すことが想定される。モニター系と LLRF 系の一部から、適用作業を進めている。

7. 課題

導入してから、いくつかの課題が見えてきたので、ここに示す。

7.1 属性の名称ルール

識別子毎に取り扱う属性を登録するシステムとした。これは制御系メッセージの英語第 5 文型(SVOC)で、識別子を O、属性を C に割り振ることをおおよそ想定していた。ところが使いだすと、'current', 'clock', 'offset'といった意味を示すものから'mm','msec','kV'といった単位系、さらに単に値の箱を表す'fvalue','ivalue'をリストに加える必要があった。後者の場合の多くは O に C の意味がすでに含まれている。また、'dB'を整数値として使ってしまったので、実数値は'dB_float'として区別するといった必要もあった。OC のリストが加速器建設段階から共有されていると見通しが良くなると思われる。

7.2 データ不整合の可能性

例えば電磁石の設定について、軌道解析の立場からの見方と、電源機器からの見方は異なる。電磁石の強さと電流値の換算係数が仲立ちとなるが、毎回一方から計算するか、常に両者を同時に更新することをアプリケーションで担保するかが不整合を起こさないために必要である。

7.3 ID-ID loop

識別子(ID)どうしの関係を1対1で定義している。例えば、element_id と eqip_id の対応は加速器の構成要素と機器の関係を定義し、element_id と magtype_id の対応は電磁石タイプを紐づける。システム上は何階層も作成でき、ループを排除する仕組みがない。安全のために登録の際のライブラリアンはおく必要がある。

7.4 部分的なグループ再編成

運転パラメータの階層構造を再編成し途中に分岐を

作ることが難しい。

7.5 二つ目の作業エリア

作業エリアの 2 つ目があっても良かったかもしれない。その場合はオフラインで過去の設定を読み返す場合、現在の設定値を変更せずに 2 つ目の作業エリアを使うことができる。今の仕様では、保存エリアにアクセスする限られた関数を利用するしかない。

8. まとめ

SPring-8/SACLA 加速器のためのアップグレード計画に向けての制御系改修のタイミングで、機器の設定値、校正値などの加速器運転パラメータ管理システムの開発を行った。SPring-8 においては 20 年来の使用状況を整理し、運用されているアプリケーションの再利用を考慮しながら、2017 年―2018 年のメンテナンス期間を使って移行を進めた。一例として、BPM の校正値は用途毎に別れた表の形式になっていたのを、切り口の違う面には別の識別子の種類をあてがい、同じフォーマットで扱えるように変更し、管理性を高めた。制御系の大きな改修からほぼ 1 年間、運転パラメータの管理について問題なく運用できている。

参考文献

[1] R. Tanaka *et al.*, “The first operation of control system at the SPring-8 storage ring”, Proc. of ICALEPCS'97, Beijing, China, 1997, p. 1.
[2] SPring-8-II Conceptual Design Report, (2014); **http://rsc.riken.jp/pdf/SPring-8-II.pdf**
[3] T.Fukui *et al.*, “The Design of the Control System for the SACLA/SPring-8 Accelerator Complex to Use the LINAC of SACLA for a Full-Energy Injector of SPring-8”, Proceedings of IPAC2019, Melbourne, Australia (2019).
[4] T.Fukui *et al.*, “Status of the Control System for the SACLA/SPring-8 Accelerator Complex”, Proceedings of ICALEPCS 2017, Barcelona, Spain (2017).
[5] 例えば Bill Karwin,児島修(訳)「SQL アンチパターン」、オライリー・ジャパン.
[6] NetApp snapshot 機能; **https://www.networld.co.jp/product/netapp/pro_info/software/soft/data_protection/**
[7] SAP-ASE; **https://www.sap.com/japan/products/sybase-ase.html**
[8] MariaDB; **https://mariadb.org/**