会誌「情報処理」Vol.63 No.8（Aug. 2022）「デジタルプラクティスコーナー」

特集号招待論文

SQLおよびSQL/MMにおける日本からのいくつかの提案とその顛末

小寺　孝[1]　鈴木健司[2]　梶野智行[3]　土田正士[4]　山平耕作[5]　芝野耕司[6]

[1]（株）日立製作所　[2]東京国際大学　[3]（株）ヒューアップテクノロジー　[4]東京都立大学　[5]（株）日立ICTビジネスサービス　[6]東京外国語大学

本稿は成功事例を通した知識や手法の共有という論文誌デジタルプラクティスの趣旨からは外れる．ISO/IEC JTC 1/SC 32のWG 3とWG 4において筆者らが手掛けた提案の中で，ここで取り上げるものは国際規格としては必ずしも成就はしなかった．国際規格にはならなかった提案に語る価値はないだろうか．今ある規格の問題点も含めてあるべき姿を考える上でも，また将来規格を検討する上でも，成就しなかった提案とそこで行われた議論について知ることには意味がある．論文とは異なり，国際規格にならなかった提案はまるで最初からこの世に存在しなかったかのように関係者外一般の目に触れる機会もなく時間の闇に葬られる．その前に提案の内容と顛末を伝え残したいと願う次第である．本稿ではSQL/MM規格の1つの部として提案したSQL/MM HistoryおよびSQL規格の追加機能として提案したSQL/Security，Hash型名前付きデータを取り上げる．また，JIS SQLに関連して行った提案および次期SQL規格への提案についても触れる．

1. 本稿で伝えたいこと

埋没していた学説が後世の目に触れてときに発掘されることもある学術的な世界とは異なり，標準化の世界では規格にならなかった提案や考えが見直されることはほとんどない．規格というものが現在の世の中で活用されるためのものである以上，それは当然ともいえる．しかしながら，規格のいくつかの案が併存していた場合，あるいは規格化のぜひそのものが議論された場合，規格として採用された案，規格化したかどうかの結果が最適であったかどうかは，並行世界で複数の決定を同時適用して評価でもしない限り知る由もなく，多くの場合不可知な問題である．ただ，今ある規格を理解する上で，また将来規格を検討する上で，規格というものが演繹の帰結のようなものでなく，いろいろな議論や考えの葛藤と交錯の過程の結果であることが一般の目に触れることの意義は小さくない．

本稿では，ISO/IEC JTC 1/SC 32のWG 3とWG 4において2000年以降に日本から提案したSQL/MM History，SQL/Security，Hash型名前付きデータを取り上げる．SQL/MM Historyは技術仕様という規格とは別の形で刊行された．SQL/Security，Hash型名前付きデータは，規格の中に取り入れられるには至らなかった．これらの提案について，機能の内容および行われた議論について伝える．加えて，JIS SQLが国際規格に果たした貢献と，現状のSQL国際規格で解決が望まれる問題についても触れる．

2. SQL/MM History

SQL/MM Historyはかつて国際規格として開発が進められたデータベースの一機能であり，日本が提案した．2005年にベルリンで行われたISO/IEC JTC 1/SC 32の総会においてであった．いささか頭字語の羅列が目立つが，JTC 1というのはISOとIECという2つの国際標準化組織が共同して情報技術の標準化を行うための下位組織であり，SC 32はその中でデータ管理の分野を受け持つ分科委員会（Sub-Committee）である．SQL/MM（SQL Multimedia and Application Packages）はデータベース言語SQLの応用分野に応じたアプリケーションパッケージの規格群（ISO/IEC 13249規格群）であり，SC 32の作業部会WG 4（Working Group 4）で開発を行っていた．アプリケーションパッケージというのは言語の応用機能のためのライブラリに相当し，言語の上の層にある．言語によっては基本機能の一部をライブラリという形で提供する場合があり，それは言語仕様には含まれないが，言語と同じ層にある言語機能の一部と考えられる（ここから先はライブラリという言葉で言語機能の上の層のライブラリを示すことにする）．

データベースのデータを更新すると，データの値は新しい値に置き換わり元の値はなくなってしまう．元の値を消さずにどこか後で見られる場所に履歴として残そうというのは，自然な要求であり，個々のデータベースのアプリケーションプログラムがそれぞれ自力で行っていた．そのための標準化されたインタフェースというのも至極当然に望まれるものであり，その提案にはあまり反対の余地のないものと思われた．しかし，思いもよらずこれが，長い論争の始まりになった．データベース言語SQLを開発するSC 32のWG 3の米国メンバを中心とする人々が反対を始めたのである．

彼らの主張は，履歴データベースの機能はライブラリのレベルではなく，SQL言語レベルで実現するべきであるということにあった．本来ならば，言語かライブラリかという議論は最初になされて当然である．しかし，履歴データベースについては，言語レベルで実現しようとして頓挫しその後新しい提案の目処もないという歴史的経緯があった．そこでライブラリレベルでの実現を考えるのは合理的な選択である．建設的な対案もなく不合理ですらあるこの反対の背景を考えるためには，時間をさらに遡る必要がある．

2.1 TSQL2

1990年代に時制データベースの研究が米国で盛んに行われ，SQLデータベースに時制の機能を取り込むための1992年版のSQL規格（通称はSQL92，開発コード名であるSQL2という呼び方も当時はよく用いられた）の言語拡張が考えられた．この言語仕様はTSQL2と呼ばれた[1][2].

1987年に国際規格として最初の版が発行されたSQL（ISO/IEC 9075）が暗に想定するデータモデルは関係モデルといわれるものである．データベースにデータを格納したり取り出したりするひとまとまりのデータ単位はレコード，あるいはより概念的なレベルでは実体と呼ばれ，データ項目とか属性という基本データ要素で構成される．レコードを行，データ項目を列とする表がSQLでのデータベースであり，データモデルにおいて表に対応するものが関係（リレーション）と呼ばれるものである（本当は，関係には数学的に歴とした定義があるが，ここではそこには踏み込まない）．関係モデルのためのデータベース言語としてのSQLはSQL92でほぼ大成した．

SQLでは表が1個のデータベースであるが，TSQL2では個々の時点の表（スナップショット☆1）を時間軸に沿って集めたさらに大きな集団を1個のデータベースと見なす．データモデルの次元拡張のようなことが必要になるが，これに対応するためには，SQLの問合せ言語，データ操作言語，データ定義言語という広範な範囲にわたるとてつもなく大きな変更が必要になる．また，データベースを構築するときには，まず概念的なレベルで実体および実体間の関連をER図という形に書き下すが，データモデルの根幹が拡張されるとき，これをどう扱うのかということも大きな問題であり，影響はデータベース言語にとどまらなかった．

SQL規格が次に目指したのは，当時ソフトウェアの世界全体で大きな潮流となっていたオブジェクト指向機能を取り込むことであり，それに加えて時制の機能を開発する余裕はなかった．TSQL2は実際に規格化提案されるには至らなかった．

TSQL2の規格化という目的は潰えたが，後の世に何も残さなかったかというとそんなことはない．TSQL2は広く時制にかかわる内容を含んでいて，履歴に関連する機能としてはデータベースの更新履歴のほかに適用時間☆2（application time）歴というものがある．これはデータベース上の履歴ではなく，実世界上での情報の変更の適用時間を扱うための機能である．たとえば，4月1日付の人事異動発令と同期してデータベースを更新する代わりに，人事データとその適用開始日を事前にデータベースに登録する運用もある．ただ，データベース上の履歴と適用履歴のための2つの時間軸が絡まると途端に複雑さが増すので，データベース機能としてはデータベース上の履歴に一本化することが現実には多い．

2.2 SQL/Temporal

同じころ，欧州でも時制データベースの研究が行われていた[3]．こちらは時制論理に的を絞ってSQLの言語拡張を行おうとする試みであり，英国から規格化の提案があった．SQL規格の1つの部（ISO/IEC 9075-7）[4]として開発されることになった．期間を扱うためのデータ型と，期間データ型を基礎にする時制論理のための一群の関数，述語，問合せ機能を提供する．

この提案に対して，時制論理だけだと実用性に乏しいという理由で，米国が反対を繰り返し，結局，開発は取り下げられた．

2.3 SQL/MM History

数年の後，日本はISO/IEC 13249の第7部としてSQL/MM Historyを提案した[5]．TSQL2では時点ごとの各表データが1個の履歴なのに対して，SQL/MM Historyでは，表に履歴の期間のための列を持たせて，表の中の1個の行で1個の履歴を扱う．行が更新されるときに更新前の行を

消さずに残して，更新後の行を表に追加するという単純な考え方である[6]．これは特に斬新な発想というわけではなく，むしろ多くのアプリケーションプログラムが行っていたことであった．しかし，似たような仕組みを個々のアプリケーションが別々に作るのはあまり生産的でない．アプリケーションプログラムの可搬性という点においても，プログラマの教育コストという点においても，アプリケーションプログラムが行うこのような処理はライブラリの中に押し込んで，その呼び出し方の標準インタフェースを規定することの利点は大きい．既存のアプリケーションはその利点を享受できないが，履歴を必要とする業務が多く見込まれ，新しいアプリケーションでの需要は引き続き大きかった．

履歴の機能を応用ライブラリで実現するということは，履歴をSQLデータベース機能の利用の仕方の1つと捉える考え方である．SQLアプリケーションはあるデータを履歴として意味付けするが，SQL処理系は履歴としてのデータの意味は知らない．一方，履歴機能を言語仕様の中で実現するTSQL2ではSQL処理系が特定のデータを履歴として認識して機能を実現する．TSQL2とSQL/MM Historyは，言語とライブラリという対極にあり，SQL/Temporalは，SQL処理系はデータに履歴としての意味付けはしないが，時制を扱うためのデータ型と時制論理のための操作を提供するという中間的な立ち位置にあるといえる．

言語かライブラリかというような二面対立的な論争は得てして果てしなく繰り返される．SQL/MM HistoryのNP（新規作業項目提案）は承認され，開発が粛々と進められたが，その一方で，建設的な対案も提示されず，理不尽とも感じられる反対の横槍が入る状況が続いた．この理不尽さの原因は結局のところはっきりとは分からない．自分たちが実現できなかったことを他人に実現されたくないという感情論とも感じられたが，Not Invented in USは認めない一種のNIH症候群と揶揄する国もあった．すべての道は言語に通ずという言語万能を崇める技術的信仰によるものだけだった可能性もなくはない．

その間欧州の一部の国でメンバの世代交代が進み，これがSQL/MM Historyにとってはあまり好ましいことではなかった．かつては，SC 32の幹事国であり最有力国である米国に対して忖度せず独自の考えを述べる尊敬するに足る国が追従者へと変質していった．そうして反対国が増えた結果，開発を粛々と進めるという状況ではなくなり，規格開発におけるCD（委員会原案）という段階からFCD（最終委員会原案）に進めるのが困難な状況になっていった．

2.4　SQLのシステムバージョン表

2008年に米国がWG 3でSQLの言語機能としてシステムバージョン表を提案した[7]．これは，SQL/MM Historyの考えをSQLの言語上に転用したものであった．そればかりか，その提案文書の中の履歴機能の使用例がSQL/MM Historyの文書の使用例と酷似する部分もあり，そういう問題に斬り込むことも可能ではあったが，協議の結果，SQL/MM History のFCD化に関して米国は反対しないということになった．全面対決ということではなく，無血開城のような結末であった．米国はシステムバージョン表によってSQL/MM History不要論を主張しSQL/MM Historyの取り下げを目論んだのであろうが，皮肉にも目論見に反する結果となった．ただ，SQL/MM Historyをさらに進めてIS（国際規格）にするということは難しく，SQL/MM HistoryをISではなく，TS（技術仕様書）[8]にすることで，2011年に最終的に決着した．SC

32の中の有力国の反対という逆風の中で形を残せたことは大きな成果であった．システムバージョン表の方は，SQLの2011年の版の中に組み込まれた．さらに，システムバージョン表とは別に適用時間の機能も少し遅れて提案され追加された．

2.5　SQL/MM Historyとシステムバージョン表の顕著な差異

SQLのシステムバージョン表は，SQL/MM Historyの考えを流用はしたが，流用できなかった重要な相違点が2つある．

1つはSQL/MM Historyでは履歴の対象となる表と，履歴の蓄積場所となる履歴表は別の表であるが，システムバージョン表は履歴対象表自体に履歴を蓄積するという点である．現実には履歴は業務系システムのマスタ表自体に保持はせず，分析系システムに置くことも多く，履歴表を別表にできないことは大きな制約になる．実際に，システムバージョン表を実装しているといわれる主要なDBMS（データベース管理システム）☆3は，履歴表を別表として対応付けの定義を行うことが必要とされ，規格とは異なる実装となっている．

もう1つの差異は，SQL/MM Historyでは履歴対象表の特定の列だけの履歴表を作成できるが，システムバージョン表は，必然的に履歴対象表自体に履歴の期間情報を追加した構成になってしまう点である．しかも，SQL/MM Historyでは同じ履歴対象表の異なる列に対する複数の履歴表を作成できる．履歴という点ではこの違いはかなり本質的であり決定的である．システムバージョン表の履歴は行の変更履歴であり，行のどの列が更新されても履歴行が生成される．一方のSQL/MM Historyでは，特定の列の更新履歴が作成でき，特定の列の変更歴を追跡することが可能となる．行の履歴には複数の列の履歴がもつれ合って存在する．もつれのない形にするために行の履歴を各列の履歴に分離する履歴の正規化が，SQL/MM Historyでは可能なのである．関係モデルでの正規化は，主キー（個々の行を一意に識別する値を持つ列の集まりのこと）を介して結び付けられた複数の事実を切り離して，データの重複や不整合の発生を抑止するために行われ，その究極の形が第5正規形と呼ばれるものである．履歴表の正規形を，第5正規形を既約な形に分解する第6の正規形[9]とする考えもある（履歴を含まない場合が特殊ケースとしてあり得るが，あくまで分解の必要性は履歴という文脈においてあると考えられ，関係モデルでの第6正規形とまでいうのは言い過ぎの感がある）．

3. SQL/Security

3.1　SQL99以降の状況

SQLの国際規格開発において，オブジェクト指向機能の取り込みは一大事業であり，1999年の版においてなされた．この版の国際規格はSQL:1999（通称SQL99）と呼ばれ，規格として承認されたのは日本の松江で開催されたSC 32総会においてであった．このオブジェクト指向機能は主に，SQL/MMの全文検索（ISO/IEC 13249-2 SQL/MM FullText）や空間情報（ISO/IEC 13249-3 SQL/MM Spatial）の利用者に使用された．これはSQL/MMというライブラリを通しての使用であり，利用者が自前で利用者定義型☆4を定義して使用するという使い方は実際にはほとんど行われなかった．

このSQL:1999発行前後のSQLの国際規格の開発には，SQLの言語仕様の共同開発という様相があった．規格としての言語仕様の開発と並行して，参加国のDBMSでその実装を行うことで，国際規格の発行と間を置かずにそれを実装するDBMSを市場投入することができた．規格の発行が実装より遅れ過ぎると，DBMSごとの独自仕様が蔓延しやすくなる．逆に規格の発行が実装に先行し過ぎるとアプリケーションプログラムでの機能の組み込みが進んで結局規格仕様が用いられずにすたれてしまう．規格発行と実装の時間差を縮めるために，当時のSQL国際規格開発の枠組みはうまく働いており，その中心に主要DBMSを抱える米国があった．オブジェクト指向機能の次にはデータ分析で必要とされる集計機能の強化（SQL/OLAP）やXML連携機能（SQL/XML）等の一連の機能が追加され，SQL規格は成熟期を迎えた．成熟するまでは各国が同じ方向を目指していたが，いったん成熟した後は，向きに乱れが生じ始め，それまでのようには足並みは揃わなかった．

3.2　SQLインジェクションとSQL/Security

規格の成熟とともにSQLデータベースが多くの業務システムの基幹として用いられ重要データを扱うことが著しく増えた．これはSQLデータベースのセキュリティ上の重要性が高まったことを意味する．実際にSQLインジェクションという攻撃による情報漏えい被害が大きく増加することになった．このような中で2009年に日本はSQLインジェクション攻撃への対策を念頭においたSQL/Securityを提案した．

SQLを使用する多くのWebアプリケーションでは入力フォームへの入力データをSQLの定数（リテラル）として表現し，この定数テキストをSQL文テキストにはめ込んだSQL文を実行する．このような仕組のもとでは，本来の入力データに，OR '1'='1'のようなテキストを付加した入力を行うと，恒真条件で問合せが行われるようになる場合がある．たとえば，入力フォーム内の入力欄に入力したデータに対する等条件が問合せの条件となる場合を考えてみる．いま，InputDataというデータを入力すると，この入力欄のデータを文字列定数'InputData'として表現し，問合せ条件は次のような形になる．

列名='InputData'

ここで入力欄に細工を施したデータInputData' OR '1'='1を入力すると問合せ条件は

列名= 'InputData' OR '1'='1'

のように改変され全件の問合せが行われてしまう．このような，入力値を通してSQL文テキストに余分なテキストを注入して，本来の意図に反したSQL文の動作を引き起こす攻撃手法をSQLインジェクションという．SQL文テキストへの余分な条件の注入自体を抑止する仕組みは，アプリケーション側に組み込むが，一般に1つのデータベースシステム上にはいくつかのアプリケーションが稼働し，1つのアプリケーションにおいてSQLを組み立てる場所も複数あるため，対策するべき個所は多い．その中のわずかな対策漏れをついて突破されたときに被害をより少なくするための機能をデータベースシステム側で提供することが望ましい．SQL/Securityはそのための2つの機能からなる．

1つはデータベースへのアクセスの証跡を保持する機能[10]である．SQL処理系が問合せの内容から悪意を検出することは難しく，アクセスの証跡を保存して監視，追跡して攻撃を察知することが必要になる．昨今の技術では，攻撃に特徴的な構文パターンを学習して攻撃を検知することも考えられるが，そのためにもSQL文を保持して蓄積しておくことが必要になる．このように証跡を用いて，攻撃の事実を早期に検出することが被害の拡大を抑えるために重要である．

もう1つは問合せの結果行数を制限する機能[11]である．これは攻撃に対する被害をより小さく抑えるダメージ制御を可能とする．問合せの結果行数が問合せの仕様によって決まることは明らかであり，現に結果行数を問合せの構文で直接指定する機能がある．しかし，構文そのものを改変するSQLインジェクションに対して構文での対策は脆弱である．終端にセミコロンを付加したテキストを注入すれば，そこまでで文を打ち切って1つの文にすることができる．構文上正しい問合せ文になるようにセミコロンの前の部分を調節することは可能である．そういう小細工が効かないようにするためには，結果行数の上限をSQLのセション属性として設定可能にするのが1つの現実解と考えられる．

これらの提案に対するSC 32/WG 3の反応は冷淡であった．そもそもが，セキュリティ問題に関しては消極姿勢であった．セキュリティ問題は可能な限りSQLの中には持ち込まず，SQLの外で行う．これがSC 32/WG 3の米国と周辺国の考えだった．提案は拒絶された．セキュリティ対策は広い視野で考える必要がある．だからと言って，SQLは関知せずに，外側で勝手にやればよいということにはならない．セキュリティ対策の中で使用できる機能としてSQLが提供するべき機能もあるはずである．

4. SQLでのHash型名前付きデータ

4.1 必要性

ここでのHash型は概念的には連想配列といってもよい．配列とは値を要素として並べたものであり，通常の配列は添字と呼ばれる要素番号によって配列要素を指し示すが，連想配列では配列要素をキー値と対応付けてキー値によって指し示す．対応付けという点からMap型ということもある．キー値は順序番号であってもよいので，通常の配列は連想配列の特殊形と考えることができる．

大規模分散環境で行われるデータ管理が多くなるとともに，連想配列形式のデータ格納法が重宝され，データの形式が柔軟でトランザクション一貫性の制約も緩い非SQLインタフェースで使用されてきた．キー値と要素値の対を格納するということで，KVS（Key-Value Store）という用語が用いられることが多い．KVSで扱うような問題は，大規模分散環境やインターネット上でのデータ基盤が整ってきて出てきた新しい問題と考えられることが多いが，SQLアプリケーションにとっては，実は新しくて古い問題でもある．その1つの典型としてコード変換表というものがある．コードや番号を対応する値，たとえば文字列の名称に変換する場合や，同じ対象に対して異なるコードを持つ複数のシステムを扱うような場合，あるいはそのようなシステムを統合するような場合に使用される．これはSQLの表として実に自然に実現できる．変換元のコードと変

換先のコードの2つの列で構成される表を作成すればよい．コード変換表の名前をcodeMap，変換元のコード，変換先の値の列の名前をそれぞれscode，dvalueとすると，コード変換表はたとえば

```
CREATE TABLE codeMap (
  scode CHAR (5) PRIMARY KEY,
  dvalue CHAR VARYING (32))
```

のような定義の表として実現できる（CHAR (5)は長さ5の固定長文字列型，CHAR VARYING (32)は最大長32の可変長文字列型のデータ型を示す）．SQLの問合せの中でコードを変換するためには，副問合せという問合せの入れ子を書く必要がある．

'コード1'という変換元コードの変換先の値を求めるためには，

```
(SELECT dvalue FROM codeMap WHERE scode='コード1')
```

という副問合せを書く（「scodeという列の値が'コード1'の行をcodeMapという表の中から求めて，その行のdvalueという列の値を結果として返す」ことを行う副問合せであるが，'コード1'のような直接的なデータ記述は変数にするなどして実際には可変にする）．一般的にコード変換表というものは数が多くてこれをSQLの表にすると2列の小さい表を山のように多数定義してこれを管理することになる．また，実際のコード変換のためには先程の副問合せを問合せの中に記述する．表の管理と問合せの記述の煩雑さとともに，問合せの実行効率が問題になることもあり，コード変換表をSQLの表にはせずに，アプリケーションプログラムの中で実現することも行われてきた．コード変換表をSQLの外側でプログラム言語のデータ構造で実現することもあるし，近年では簡易なKVSを使うという選択肢もあるだろう．

ただ，共用データ構造をアプリケーションプログラムからできる限り独立にし，生産性や保守性を高めるというデータベースというものが生まれてきたそもそもの由来と存在意義を考えると，コード変換表もSQLの中で扱えるべきである．

4.2　Hash型名前付きデータの提案と顛末

キーと値の対をSQLの中で簡易に扱うために日本から2015年に，連想配列を扱うためのデータ型としてのHash型と，表からは独立した名前付きデータの機能を提案した[12]．この機能では，たとえば先程のコード変換表は

```
CREATE DATA codeMap CHAR VARYING (32) HASH[CHAR (5)]
```

のようなCHAR (5)（長さ5の固定長文字列型）のデータ型のキーとCHAR VARYING (32)（最大長32の可変長文字列型）のデータ型の値の対の集まりを保持するためのHash型のcodeMapという名前を付けたデータとして実現できる．キーを変換元コード，値を変換先とすれば，SQL文の中では，codeMap['コード1']という式によって'コード1'に対応する変換先の値が得られる．

名前付きデータ自体はHash型専用というわけでもない．SQLにはマクロのような機能がないので，同じ値を多くの場所で使用する場合には，同じ値を定数として多くの個所に直書きするしかない．そうしたくない場合は，SQLの外側のプログラムの変数に値を設定したり，コード変換表まがいの小さい表を定義したりということになる．名前付きデータの副産物として，SQLのこの不便さを改善することも期待できる．たとえば予算の上限値がいろいろな個所で参照される場合，次のようなDEC (10)のデータ型の予算上限という名前のデータを定義する．

CREATE DATA 予算上限 DEC (10)

この予算上限に

SET 予算上限 = 100000.

のように値を設定しておけば，多くのSQL文に同じ値を直書きしなくても予算上限と書くだけで済む．年度が変わって予算の上限値が変更になっても，このデータを更新すれば参照するSQL文は修正する必要もないし，マクロの記述を変更したときのような再コンパイルも必要ない．

これを提案したときには，反対なく受け入れられたように見えた．ISO/IEC 9075規格群（データベース言語SQL）のプロジェクトエディタからは，SQLの可能性が高まったと謝意まで述べられた．ただし，Hash型という名称はMap型に変更するということになった．HashからMapへの大修正をして半年後の会議に臨んだが，会議の直前に米国で反対が起きた．SQLに表以外のデータ構造を持ち込むべきではないということであった[13]．200ページ近い提案寄書を書き上げた挙げ句に梯子を外されたかのように状況は様変わりした．

SQLデータベースで基本となるデータ構造は表である．そこに異論の余地はない．名前付きデータが表と対等に扱われるものならば，排除されてしかるべきであろう．しかし，名前付きデータはあくまで表の外にある補助的なデータ構造として使用されるもので，データベースというより永続化（削除しない限り揮発したりしないということ）された共用データである．それを異物と見なして排除するのは，過剰な免疫反応であり，多分に一神教的でさえある．唯一の神に逆らうのは誰しも怖い．議長は米国出身ながら機能の必要性については擁護的であったが，名前付きデータは異物であり，そのためにCREATE DATAのような定義文を増やすべきではなく，論理的に同等なことを表によって実現するべきである，というのが米国とその周辺国の考えであった．もはやこの提案に賛成する国はなかった．数十年に渡るSQLの長い歴史の中で侵されなかったタブーに触れてしまったのだろうか．SQLにKVSを持ち込む意図であった，と言い捨てて会議での議論を筆者は終えた．

4.3　SQL/MMでの提案

SQL機能としてのこの提案は却下されたが，機能の必要性自体を全否定されたわけでもなかった．表向きはHash型の名前付きデータに対するのと同等な操作でも，それを表への操作に構文変換するようなやり方ならばよいということであった．次はそのとおりに，SQL/MMの第9部（ISO/IEC 13942-9）UHASH[14]として提案した．SQL/MMでは，アプリケーションプログラムが機能を呼び出すためのSQL記述が規格仕様に適合してさえいれば，実装を規格の記述とおりに行わなければならないという制約はない．したがって，表に基づいて仕様の記述を行って

も，実装は簡易なデータ構造で等価な仕様を実現するということが可能である．しかし，今度は規格開発に必要な人的資源がないという理由でSC 32で却下された．ここでSQLにHashデータを持ち込むという試みは潰えた．

この時期，SQL/MMでは第9部のほかにデータ分析にかかわる2つの部を提案した．第10部（ISO/IEC 13942-10）変化検知[15]と第11部（ISO/IEC 13942-11）深層学習[16]である．データベースのデータを使用して機械学習を行うために，大量のデータをPython等の実行環境に搬出せずに，データベース側で実行する需要があるのは明白であった．第10部 変化検知は，時系列データの変化度を計算する機械学習手法を用いて特異変化を検知するためのデータベースの定義，モデルの学習，評価，適用のための関数を生成するためのインタフェース群を提供する．第11部 深層学習は，すでに多く利用されているPython等の言語上の深層学習用ライブラリに対応するSQLのライブラリを提供するものである．これらの機能の必要性は認められたが，第9部と同じような理由で却下された．

反対理由を技術上の思想の問題に帰着させようとするとき，それは実際には表向きのことで，背後に切実な別の問題があることがある．米国のデータベース言語の規格開発を担当するANSI X3H2では参加企業が2000年以降減少の一途を辿った．グラフ問合せ言語の開発が行われるようになって以降は盛り返しをみせているが，SQL規格の開発がごく限られた人員で行われている状況は変わらない．さらにその先にはDBMSでのSQLの実装がある．グローバルネット企業でのサービスや分析のために使用されるSQLの開発，またOSSでのSQLの開発が精力的に行われる反面，データベースのベンダでは開発の主軸がDBMSの外側に向かいSQL自体の開発はかつてのように行われているようには見えない（規格開発に携わる人員の少なさはその裏返しと考えられる）．データベース機能の中核に手を加えて，いまさら名前付きデータのようなものを組み込む余力はないのかもしれない．規格開発のためにも，実装のためにも投入する資源がないということと推測される[17]．

5. SQLのISO規格とJIS規格

SQLのISO規格の初版以来，対応する日本国内の一致規格としてJIS規格（JIS X 3005規格群）が作成されてきた．規格として国際規格があればよいと考える向きもあるが，日本語の規格の存在は，規格仕様のより正確な理解が国内でより広く行われるためにきわめて重要であり，日本語の共通用語を提供することは，SQLにかかわる人と人とのコミュニケーションを円滑に行うために必要である．SQLにかかわる産業への貢献のみならず，国家試験での参照規格としても使用され，人材育成や教育上の役割も大きい．

SQL国際規格は版を重ねるごとに巨大化し，1999年の版で複数部構成となり，現在では4千ページを超える．2000年以降の版でも，主要な部は全訳JISを作成したが，肥大化したISOの原規格には多くの記述の不備があり，技術的不備も散見された．日本はJISの原案作成作業で発見されたそのような不備の修正提案を行った．一つひとつの提案の規模は小さいが，1回の会議で数百件に上ることもあり骨の折れる作業であった．品質向上という作業は実に地味ではあるが，SQL処理系としてのDBMS，SQLを利用するアプリケーションプログラムの信頼性を向上させるだけではなく，開発における混乱やトラブルが低減されることには，実は生産コストにも大きな

寄与がある．このような提案には反対もなく歓迎された．規格が円滑にそして正しく適用されるためにその品質向上は不可欠であり，この点におけるSQL規格への日本の貢献は，国際規格開発参加国の中でも特別に大きかった．

6. SQL次期規格

SQL次期規格でのSC 32/WG 3の関心は，SQLの表の上にグラフ構造を構築してグラフ問合せを行うための新しい部（ISO/IEC 9075-16）SQL/PGQ（Property Graph Query）とそれに対応するグラフ問合せ言語GQLだけに注がれている．次期規格ではSQL/PGQを含む主要な部を改正するが，このCD（委員会原案）投票において，日本はあえてSQLの既存部分に対していくつかのコメントを提出した．

SQLはデータベース言語として多くの情報システムの深層で使用されており，この状況は今後も当面変わることはないと考えられる．一方，2003年の版で導入されたOLAP機能を中心とする集計にかかわる豊富な機能のデータ分析での利用が広がっている．データベース言語でありながら，分析用言語としての性格も併せ持つSQLは二重人格的といえる．そのような言い方はいささか聞こえが悪いので，二刀流という方がよいかもしれないが，二面あわせ持つことによる難しさも当然ある．日本が提出したコメントのうちこれに関連して大きな問題と思われるものについて述べておく．

6.1 トランザクションの隔離性水準

データベース言語と分析言語ではトランザクションに求められる要件は異なる．データベース言語では更新に対する整合性がきわめて重要である一方，分析言語では大量のデータの参照が迅速に行えることがより重要である．これらは別々に実現されるわけではなく，同時に実行される可能性がある以上，相互に整合性を保つことが必要である．

トランザクションの隔離性水準は，同じ表に読み書きを行う複数のトランザクションを同時実行した際の相互作用によるデータ整合性の低下の程度を表す．トランザクションを直列に並べてそれぞれ単一で実行した場合と同時実行した場合で同じ実行結果になれば整合性は相互に維持されている（これを直列化可能という）．相互作用の結果として実行結果が異なってくると整合性に乱れが出てくる．整合性の乱れは，それがもたらす特徴的な現象によっていくつかの水準に分類され，これが隔離性水準である．SQL規格の隔離性水準は，単一版のロック法という同時実行制御で発生する現象だけに基づいている．ここでいう版はデータが更新されたときに生成される版のことであり（物理的な格納データの版であり，SQLの表の中で論理的に意識される履歴としての版のことではない），単一版ということは，データが更新されても新しい版は生成されず，更新前のデータは上書きされて残されないということである．

現在のトランザクション同時実行制御では多版が主流になってきている．多版同時実行制御[☆5]はロックを用いないと誤解されることが多いが，単一版制御にロック法に基づく方法や時刻印を用いる方法があるように，多版制御にもロック法に基づく方法（多版であることによってロックを必要とするアクセスの競合は減る）や時刻印を用いる方法がある．実際のDBMSで多く用いられる多版制御は時刻印法とロック法の混合法に近いスナップショット隔離というものである

[18]．直列化可能な形でこの方式を実現しようとすると，ロックとは別の代償（直列化可能性を阻害する状況が検知された場合のトランザクションの取り消し）が必要になる．そのため，直列化可能性を緩めたやり方が用いられることが多く，書き込みのねじれ☆6（write skew）といわれる不整合現象が発生し得る．データの書き込みを行わないトランザクションの集まりが同時実行される限りにおいては，このような現象は起こらない．データ分析を行うためのアプリケーションでは，そのような状況もあり得る一方，基幹データを扱うデータベースアプリケーションでは，一般にそのような理想的な状況にはならない．この不整合現象を扱わないSQL規格は現在のDBMSのトランザクション制御に対応できていないといえる．

6.2　役割に基づくアクセス制御

SQLの表にアクセスするためには認証と認可が必要となる．認証はSQLシステムの利用者としての正当性を確認するものであり，利用者識別子（ユーザ名）に対して行われる．認可は個々の表やその中の特定の列へのアクセスの許可のことであり，認可識別子に対する権限付与という形で行われる．古い版のSQLでは利用者識別子がそのまま認可識別子として使用されていた．本来，認証のための利用者識別子と認可識別子は別の概念であるが，この2つが同じであることに不都合はあるだろうか．

1人の利用者は常に同じ役割でSQLを実行するとは限らない．利用者識別子を認可識別子として用いる場合，その利用者が担うすべての役割で必要な権限すべてを同じ認可識別子に付与するしかない．たとえば，個人の購買歴や行動歴のようなデータを保持する表を考えてみる．この表は，分析者がデータの集計等を行うために使用し，個人を区別するための識別情報を含むが，分析という目的上，実世界の個人を特定するために利用可能な氏名や住所のような情報は含まれないか，もしくは匿名化されているものとする．一方，個人の情報を管理する別の表に個人の識別情報と氏名等の属性情報が保持されていて，この表は情報管理者が管理しデータ操作を行うものとする．ある1人の利用者が，分析者であると同時に情報管理者でもある場合，その認可識別子は両方の仕事のためのデータ参照や操作に必要な権限を持たないと，片方の仕事しかできない．問題はこの利用者の利用者識別子に，両方のデータ参照や操作に必要な権限を与えてしまうと，それらの権限が同時に使えるため，SQLで2つの表の結合を行うことにより，データを連結し個人とデータを突き合わせることがいとも簡単にできてしまうことである．データが他人に知られたくないプライバシーにかかわる情報を含んでいる場合，これは実社会において深刻な問題を引き起こすおそれがある．

本来アクセス権限は利用者ではなく役割に対して与えられるべきものである．アクセス権限を役割に対してだけ与えると，分析者と情報管理者という2つの役割を持つ利用者でも，分析者という役割でSQLを実行するときには分析者の持つ権限でしか表にアクセスできず，情報管理者という役割でSQLを実行するときには情報管理者の持つ権限でしか表にアクセスできないという状況にすることができる．そうするとどちらの役割も片方の表にしか権限を持たないので2つの表を結合することができない．このように役割名を認可識別子として用い，役割に権限を付与することによってアクセスを制御する手法を，役割に基づくアクセス制御（RBAC：Role-Based Access Control）という[19]．

SQL規格では1999年の版で役割が導入され，RBACを実現できるようになった．ところが，2003年の版ではなぜか，利用者がSQLを実行するとき，どのような役割で実行するかにかかわらず，その利用者に与えられたすべての役割に付与された権限も有効になるように変更された．行き着く先は肥大化した権限を与えられた利用者のモンスター化であり，最小権限の原則とは真逆の状況となる．利用者識別子には権限を与えず，役割名に対してだけ権限を与えることによりRBACは実現できる．現在のSQL規格に役割という機能はあるが，RBACは実現できないと言わざるを得ない．1999年の版の仕様に戻すことが求められる．

7. 後記

ISO刊行物の中にISO/IEC 13249-7（SQL/MM History）は今はもうない．SC 32の2021年6月の総会でTSの取り下げが決定された．米国の発案であった．米国の発案寄書で取り下げ理由の1つに挙げられていたSQLのシステムバージョン表で機能的に十分という点に関して，日本は反論の寄書を提出したが，総会での議論はその問題に深入りすることなく，淡々と処理された．規格に代表される標準化刊行物は世の中で活用されてこそのものであり，なくなったものを知ることに意味はないといわれるとそのとおりだろう．ただ，なくなった案で取り入れようとしたこと，規格で採用された案の機能ではそれが実現できていないこと等，今の規格に欠如していることが明確になることは，規格の活用，将来の規格の検討上意味がある．消えていった提案を語ることを通してそのようなことを少しでも伝えることができたらと思う．そのような機会が与えられたことに厚く感謝する次第である．

参考文献

1）Snodgrass, R. T. 編集：The TSQL2 Temporal Query Language, The TSQL2 Language Design Committee, Springer Science & Business Media, LLC.
2）Snodgrass, R. T. et al. : TSQL2 Temporal Language Specification, ACM SIGMOD Record 23, No.1 (Mar. 1994).
3）Date, C. J., Darwen, H. and Lorentzos, N. A. : Temporal Data and the Relational Model.
4）Melton, J. : ISO/IEC JTC1/SC32 WG3 : RTM-007, ISO 9075-7 SQL/Temporal WD.
5）ISO/IEC WD 13249-7 SQL Multimedia and Application Pakages Part7: History (Nov. 2005).
6）Suzuki, K., Kajino, T., Ishii, Y., Kotera, T., Yamahira, K. and Shibano, K. : SQL/MM Approach to Temporal Database — SQL/MM Part7 : History, THE JOURNAL OF TOKYO INTERNATIONAL UNIVERSITY.
7）Kulkarni, K. : ISO/IEC JTC1/SC32 WG3 : FAO-015, System-versioned Tables.
8）ISO/IEC TS 13249-7 : Information Technology — Database languages — SQL Multimedia and Application Pakages — Part7 : History (Feb. 2013).
9）Date, C. J. : An Introduction to Database Systems (8th Edition), PEARSON Addison Wesley.
10）Tsuchida, M., Kotera, T. and Shibano, K. : ISO/IEC JTC1/SC32 WG3 : LCY-027, SQL Audit Facility.
11）Kotera, T., Tsuchida, M. and Shibano, K. : ISO/IEC JTC1/SC32 WG3 : LCY-026, Facility to Limit the Number of Rows Accessed in an SQL-Session.
12）Kotera, T., Shibano, K. and Tsuchida, M. : ISO/IEC JTC1/SC32 WG3 : NRT-028r2, MAP Data Type and Persistent Named Data.

13）Zemke, F. : ISO/IEC JTC1/SC32 WG3 : NRT-029, Response to NRT-028r1.
14）ISO/IEC JTC1/SC32 N2798 : Text of NP 13249-9, Information Technology — Database Languages — SQL Multimedia and Application Packages — Part 9 : UHASH.
15）ISO/IEC JTC1/SC32 N2799 : Text of NP 13249-10, Information Technology — Database Languages — SQL Multimedia and Application Packages — Part 10: Change Pattern Detection.
16）ISO/IEC JTC1/SC32 N2862 : Text of NP 13249-11, Information Technology — Database Languages — SQL Multimedia and Application Packages — Part 11 : Deep Learning.
17）芝野耕司：SQL言語の開発と日本の貢献，日本データベース学会功労賞記念講演.
18）Berenson, H., Bernstein, P. A., Gray, J., Melton, J., O'Neil, E. and O'Neil, P. : A Critique of ANSI SQL Isolation Levels, ACM SIGMOD Record, Vol.24, Issue 2 (May 1995).
19）ANSI INCITS 359-2004 : Role Based Access Control.

脚注

☆1　TSQL2では行の列値が更新された時にその行は更新されずに，新しい行が生成されるが，同じ1個の表に行が生成されるとは考えられない．データベースが時間的な広がりをもっていてデータベース更新時点の部分に挿入される．更新された列データと過去から値が継続される列データとを合わせてその時点の表と考えられる．個々の時点の表のことをスナップショットという．

☆2　適用時間（application timeまたはvalid time）は実世界での情報の適用の履歴に対してアプリケーションが刻印する時刻印．これに対してデータの変更に対してシステムが刻印する時刻印はsystem timeまたはtransaction timeといわれる．

☆3　DBMS／Database Management System：データベースを管理し，アプリケーションプログラムからのデータベースに対する要求を制御，実行するシステム．SQLの文脈では，SQLの記述に基づいて，データの検索，操作，定義を行う処理系．

☆4　利用者定義型 データ構造とそれに対する操作を定義するデータ型をSQLでは利用者定義型（user-defined type）という．多くのプログラム言語での抽象データ型，クラスに相当する．

☆5　トランザクションからデータベースの同じ対象に変更要求があった場合に，その対象の新しい版を生成することによって変更要求の競合を回避する手法．同じ対象に変更要求があっても別の版を変更することになるので暫定的には要求を同時に受け付けることも可能である．ただし，同時書き込みを許容してもトランザクションの終了時に変更を確定する際に，ロック法や時刻印法を用いてデータの一貫性確認を行うことは必要になる．

☆6　同時に実行される2つのトランザクションによるデータベースの異なる対象への参照と更新が交差する場合（たとえば1つのトランザクションがaを参照してbを更新し，もう1つのトランザクションがbを参照してaを更新する状況），それぞれのトランザクション内ではそれらの対象に対するデータ一貫性（たとえばa>bという制約がある場合）を維持するようにデータを更新してもこれらをあわせた全体の実行結果としては維持されないことがある（単一版でロック法を用いるとデッドロック状態になるが，多版ではそれぞれ更新できる）．異なる対象への個別の更新がそれらの間の制約にひずみを生じさせる現象

を書き込みのねじれ（write skew）という．

小寺　孝（正会員）
1987年（株）日立製作所入社．元SQL/MM Part1 Frameworkエディタ．現在，同社サービスプラットフォーム事業本部DB部所属，東京外国語大学非常勤講師，IPA情報処理技術者試験委員，本会認定情報技術者，本会情報規格調査会SC 32幹事．

鈴木健司（正会員）
NTT情報通信研究所を経て，東京国際大学人間社会学部教授，同大学副学長を歴任．データベース技術の研究実用化・国際標準化に従事．現在，東京国際大学名誉教授，電子情報通信学会フェロー，本会情報規格調査会SC 32専門委員会委員．

梶野智行（非会員）
2001年筑波大学大学院理工学研究科修了，同年，（株）ビーコンIT（現（株）ユニリタ）入社．時制DWHソフト等の開発に携わるかたわら，SQL/MMの標準化活動に参画．元SQL/MM Part 7 Historyエディタ．現在，（株）ヒューアップテクノロジーでWebアプリ開発担当．

土田正士（正会員）
1983年筑波大学大学院理工学研究科修了．同年（株）日立製作所システム開発研究所入社．DBMSの研究開発，ビッグデータ事業開発に従事．博士（情報学）．2018年国立情報学研究所を経て，現在東京都立大学ソーシャルビッグデータ研究センタ所属．本会情報規格調査会SC 32専門委員会委員長．本会フェロー．

山平耕作（非会員）
1981年九州工業大学大学院情報工学研究科修士課程修了，同年（株）日立製作所入社．DBMSの開発に携わるかたわら，SQL/MMの標準化活動に参画．SQL/MM国内幹事，JIS原案委員会幹事歴任．現在，同社OSSソリューションセンタでDBの技術支援担当．

芝野耕司（非会員）
東京外国語大学名誉教授．ISO/IEC 13249（JIS X 3006）SQL/MM国際主査，JIS原案委員会委員長，ISO/IEC JTC1/SC2 （Coded Character Set）国際委員長，JIS文字コード関係原案委員会委員長を歴任．

受付日：2022年2月22日
採録日：2022年4月19日

編集担当：上條浩一（東京工科専門職大学）