手続き型の思考と集合指向の思考

作成者：蛭田 昌伍

作成年月日：2024年6月26日

所属：システム開発部

このレポートのテーマは「手続き型の思考と集合指向の思考」です。

このテーマに設定した理由は、この「達人SQL」の演習問題を解いていくうえで、集合指向の思考をうまく行えないことが多く、苦労したため、2種類の思考を整理することで今後の負担を減らしたいと考えたからです。問題をこなすうちに多少はSQLのコーデイングに慣れてきたとはいえ、いまだにJavaのコードを記述するときと同じ感覚で思考してしまい、頭がこんがらがってしまうことが多々あります。

そこで、このレポートにおいてJavaに代表される手続き型の思考と、SQLの集合指向の思考に対して考察を行ったあと、その違いを明確にしていきます。

主に、ある程度の学習経験を積んだJavaとSQLを代表例として取り上げ、その特徴や根底にある考え方への考察を行っていきます。JavaとSQLではその性質や使用目的(方法)も異なる場合が多く、対照的な二つの思考の比較に用いるにはふさわしくないため、本文ではあくまで「手続き型の思考と集合指向の思考」の構造や、根底にある考えへの考察に用いる道具という立ち位置で取り上げています。

1.手続き型の思考

手続き型の思考は、代入・分岐・ループを基本単位として、システム全体をその基本単位に分割する発想のことです。

この思考方法の特徴は、「わかりやすさ」です。SQLに比べて(私(達)が最初に取り組んだ言語ということもありますが)、コードを見て直観的に処理内容を理解することができます。その理由はいくつか考えられます。

まず、基本的に処理の実行順序は「上から下・左から右」で固定されているという点が、コードの可読性を大きく助けています。初見のコードを見る際は、まず上から下に流し見を行い、その後で全体の代入や分岐の詳細にみていく場合がほとんどであると考えられるため、基本構造として処理が上下の順に実行されるようになっている手続き型のプログラミング言語は、集合思考のものよりも可読性が高くなるのは当然だと言えます。

次に、処理を実行順に追っていくため、プログラム全体の意図が理解しやすいという点が挙げられます。(あくまで私の認識ではありますが)プログラムは、元手となる入力から、求められる出力を得るための物なので、入力から出力までのプロセスが記述されている手続き型は、プログラムの処理構造を理解する上ではとても親切です。どの値を最初に処理するのか、処理した値はどこに格納するのか、どの時点で、なんの条件で分岐するのか、などの処理の手順が視覚的に理解しやすく記述されているため、製作者の意図が汲み取りやすくなっています。

最後に、プログラムの全体像がイメージしやすいという点が挙げられます。これは1つめの特徴を踏まえての特徴でもありますが、「実行が上から下」であることが、プログラムの「始まりと終わり」と「プログラムの規模」の把握に役立っています。私の印象ではありますが、SQLであれば小さな記述量でも大きな処理を行ている場合がありますが、Javaでは(ループは別として)基本的に記述量と処理の規模は比例していると感じました。

まとめると、手続き型の思考は、プログラムを順序立てて処理を組み立てていくため、直観的にその順序・意図・規模感が把握しやすいという利点があるということです。また、直観的にわかりやすいということは、変更を加えやすいという利点にもつながると考えられます。

ただ、わかりやすいという利点は、コードの冗長性やパフォーマンスの面では欠点となります。手続き型の指向は、複雑な処理を単純な処理に分解し、理解しやすくした反面、処理構造を冗長にし、無駄なプロセスを増やしたとも言えます。

2.集合指向の指向

集合指向の思考は、同じ性質のデータであれば、できる限り一括で処理しようという考え方です。

手続き型の思考においては、複雑な処理をできる限り単純なものに分解することで「わかりやすさ」を向上させ、直観的な対処を可能にしたのに対し、集合指向の思考においては、テーブルという複雑な対象をできる限り複雑なままで処理することで、SQLの記述を簡潔にし、処理のパフォーマンスを向上させています(もちろん処理の内容によります)。

集合指向の思考の特徴は、「わかりにくい」です。手続き型のプログラムの記述と異なり、線形に続いていく処理を組み立てるという発想ではなく、求めるデータへ絞り込むにはどのような組み立て・条件が必要かという発想で記述がなされます。イメージとしては「集合」という大きな円の中の、さらに小さな「条件にあった集合」という円を切り抜く作業です。

円のイメージでデータを扱うため、当然直観的な理解は難しくなります。手続き型のように処理を順序だてて実行することは、人間が日常の動作の中で常に行っていることなので認知しやすいですが、円(集合)を絞り込んだり、組み合わせることは、たとえ日常的に行っていたとしても目に見える形で認知することは難しいからです。

直観的理解を妨げる要因としては、処理が実行される順序が上から下ではないという点も挙げられます。上下でなく「句」によって条件が発動する順序が決まっているため、コード全体を俯瞰して見ない限り、テーブルがどのような順序で条件にかけられているか把握できません。

また、テーブルの概念も、若干理解が難しい側面があります。二次元の表を見るとついつい表にデータが並んでいるように感じますが、実際には行や列に表のような順序は存在しません。そのようなテーブルに対する認識のずれが、テーブルの結合処理の際には大きな足枷になっていると感じます。

3.二つの相違点

手続き型の思考と集合指向の思考の根本的な思想の違いは、最優先に考える対象です。

手続き型の思考の最優先は「ユーザー」です。

その理由は、手続き型の思考が用いられる場面が、アプリケーション作成などの外部スキーマなどだからです。

ユーザーが利用するシステムであれば、利便性や柔軟な対応・保守性などが重視されると考えられます。そのような場面においては「ユーザー」の立場から逆算してシステムが作られていくため、複雑な管理や変更に耐えるために「わかりやすさ」が重要であり、直観的に理解しやすい手続き型の思考が採用されているのではないかと考えます。

集合指向の思考の最優先は「データ」です。

その理由は、集合指向の思考が用いられる場面が、データベース全体の構造やデータの関係を定義する概念スキーマなどだからです。

内部スキーマで定義されたデータの構造を、データの整合性とパフォーマンスのバランスがとれた最善の形で定義する必要がある概念スキーマにおいては、「データ」を無駄なく活かすことが重要であるため、集合指向的な思考が求められたと考えます。

「達人SQL」での主題はSQLへの理解を深め、より使いこなせるようにすることでした。

今回のレポートで、SQLの根底にある「集合指向の思考」を、「手続き型の思考」と対立させて考察することで、より客観的に分析し、理解を深めることができたと感じます。

教本を通して得た学びを、今後の研修および実務で活かせるように努力していきます。

以上