DB で使用する一般的な用語

作成日: 2018/02/08
作成者: 西村

更新履歴

更新日	更新概要	作業者
2018/02/08	・　新規作成	西村
	・	
	・	
	・	
	・	

目次

更新履歴 1
はじめに 3
マスタ (マスタテーブル) 4
トランザクション (トランザクションテーブル) 4
主キー (Primary Key, PK) 5
　複合主キー 6
　サロゲートキー 7
外部キー (Forign Key, FK) 8
ユニークキー (一意キー) 9
正規化 10
　非正規化 11
トランザクション (トランザクション処理) 12
　トランザクション処理の流れ (コミット、ロールバック) 12
ロック 13
　悲観的ロック 13
　楽観的ロック 13
照合順序 (COLLATE) 14

はじめに

この資料では、主に DB で使用する一般的な用語について簡単に説明します。DB 関係でわからない用語があった場合に参考にしてください。

(ご連絡)

- RDBMS、列、行などの用語は SQL 基礎の資料を参照してください
- すぐに全部分かる必要はありません。案件によっては使わない機能や用語などもあるので、必要になった時に見て頂くとよいと思います。

マスタ (マスタテーブル)

マスタは、システムの運用上変更が頻繁にない基本的な情報が格納されたテーブルのことです。

例えば会社の情報が格納されたデータベースであれば「社員情報」「部門情報」などがマスタテーブルになり、「社員マスタ」「部門マスタ」などと呼ばれます。

マスタは「m_」や「mst_」などの名前から始める、というルールでテーブル名がつけられることがあります。(例: mst_emp, m_社員)

トランザクション (トランザクションテーブル)

トランザクションは、システムの運用上、日々の記録が格納されたテーブルのことです。

例えば会社の情報が格納されたデータベースであれば「受注情報」「業務日報」などがトランザクションテーブルになります。

トランザクションは「t_」や「tbl_」などの名前から始める、というルールでテーブル名がつけられることがあります。(例: tbl_acp, t_受注)

※ DB 処理上で別の意味の「トランザクション」という用語があります。「トランザクション (トランザクション処理)」を参照してください。

主キー (Primary Key, PK)

主キーは、簡単に説明すると「あるテーブルのデータを必ず 1 件ずつ特定できる主要な列、と決めた列」のことです。

例: 社員情報が格納されたテーブル

社員番号	姓	名
001	北海道	一郎
002	青森	次郎
003	岩手	三郎
004	宮城	四郎

例えば上のテーブルで「社員番号の列を主キーとして設定する」と決めた場合、社員番号で検索すれば必ずデータ 1 件が特定できる、という前提ができることになります。(001 = 北海道さん、002 = 青森さん…)

逆に言うと、主キーに設定された列は、「重複する値を格納することができない」という制限(制約)ができます。

社員番号	姓	名
001	北海道	一郎
002	青森	次郎
003	岩手	三郎
004	宮城	四郎
001	秋田	五郎

↑ このデータを入れようとしてもエラーになる。(001 = 北海道さんと秋田さんの 2 件になってしまうため)

複合主キー

主キーは、複数の列を組み合わせて設定することもでき、これを「複合主キー」(または単に複合キー ※1)と呼びます。

例えば社員情報が格納されたテーブルで、「姓と名の組み合わせを主キーとする」として設定することもできます。※2

社員番号	姓	名
001	北海道	一郎
002	青森	次郎
003	岩手	三郎
004	宮城	四郎

姓を主キーにした場合は、性が「北海道」さんが 2 人いると値が重複するのでエラーになってしまいますが、姓と名を主キーにした場合は同姓同名の「北海道一郎」さんがいなければエラーになりません。

※1 「複合キー」は「複数の列をキーにしたもの」という意味で、ユニークキー(後述)で複数列設定したときも複合キーと呼ばれることがあります。

※2 例としてのわかりやすさを重視して姓と名を例に出しましたが、実際のシステムでは姓や名などを主キーにするのは望ましくないです (重複する可能性が高すぎるため)。ID やコードなどの重複する可能性がないものを主キーにしてください。

サロゲートキー

主キーは他のテーブルなどからデータを参照する際に使用するため、基本的に「どんなことがあっても値が変わらないこと」が望ましいです。そのため、機械的な連番(シーケンス、AUTO_INCREMENT 等)をデータに割り当てた列を作ってそれを主キーに設定することもあり、その機械的に設けられた主キーのことを「サロゲートキー」と呼びます。

例: 社員情報に「ID」というサロゲートキー列を用意した例

ID	社員番号	姓	名
1	**001**	北海道	一郎
2	002	青森	次郎
3	003	岩手	三郎
4	004	宮城	四郎
5	**001**	秋田	五郎

この場合、社員番号が重複していてもエラーになることがありません。

この例ではメリットがないように見えますが、例えば「社員が 1000 人を超えてしまい、社員番号を 4 桁にしないといけなくなった」というようなとき、サロゲートキーがあれば社員番号の変更がしやすくなります。
(社員番号を主キーにしていると、他のテーブルなどから参照されている可能性があるので変更の影響範囲が大きくなる。重複を禁止したいだけであれば、主キー以外にユニークキー(後述)でも実現できます)

※ サロゲートキーを設けるか設けないかは一長一短あり DB 設計者の好みにもよるため、案件によってサロゲートキーがある場合と無い場合があります。

外部キー (Forign Key, FK)

「外部キー」は、簡単に言うと「別のテーブルのデータと連携するために、別テーブルのデータの列の値を格納した列」のことです。

例: 社員情報が格納されたテーブル

社員番号	姓	名	部門コード
001	北海道	一郎	**01**
002	青森	次郎	**01**
003	岩手	三郎	**02**
004	宮城	四郎	**02**

例: 部門情報が格納されたテーブル

部門コード	部門名
01	開発部
02	デザイン部
03	事務管理部

この場合、社員情報の部門コード列が外部キー列です。「北海道一郎」さんは部門コード 01、部門コード 01 は「開発部」のため、「北海道一郎さんは開発部」というデータになります。

外部キーは、データベースの機能で外部キーとして設定することで制限をかけることができます。(外部キー制約。参照先の列にない値は設定できないようにするなどを決められる)

案件によっては「外部キーとみなすけど外部キー制約の設定はつけない」という取り決めが開発者間でされることもあります。(特に小規模なプログラムの場合、制約がない方が運用しやすい時があるため)

ユニークキー (一意キー)

ユニークキーは、簡単に説明すると「あるテーブルのデータを必ず 1 件ずつ特定できる列、と決めた列」のことです。主キーも同じような意味合いを持ちますが、主キーは 1 つの列(または 1 つの組み合わせ)しか設定できません。ユニークキーはそういった制限はありません。

例: 社員情報に「ID」というサロゲートキー列を用意した例

ID	社員番号	姓	名
1	**001**	北海道	一郎
2	002	青森	次郎
3	003	岩手	三郎
4	004	宮城	四郎
5	**001**	秋田	五郎

前述のサロゲートキー列(ID)を用意した社員情報テーブルは、なにもしていないと「社員番号が被っても OK」というテーブルになってしまいます。「社員番号をユニークキーとして設定する」としておけば、重複を弾くことができます。

正規化

正規化は、簡単に言うと「データを扱いやすく・問題を少なくするためにテーブル構造を整える」ことです。

例: 部門の情報がそのまま入った社員情報

社員番号	姓	名	部門名	部門の電話番号
001	北海道	一郎	開発部	000-000-0001
002	青森	次郎	開発部	000-000-0001
003	岩手	三郎	デザイン部	000-000-0002
004	宮城	四郎	デザイン部	000-000-0002

上記の例の場合では、部門名・部門の電話番号が全部社員情報のテーブルに記録されています。これだと部門の電話番号を社員ごとに変更することもできてしまう問題があります。

この場合は、社員情報と部門情報にテーブルを分けて整えます。

例: 社員情報

社員番号	姓	名	部門コード
001	北海道	一郎	01
002	青森	次郎	01
003	岩手	三郎	02
004	宮城	四郎	02

例: 部門情報

部門コード	部門名	電話番号
01	開発部	000-000-0001
02	デザイン部	000-000-0002
03	事務管理部	000-000-0003

正規化には種類や段階があります。

正規化(整理すること)の例	整理した後の形の呼び名
・1 つの列に複数の値が入ってしまっている表を 1 件にする ・導出項目(数量×単価=合計金額の合計金額など)を削る	第一正規形
・コードなどで特定できるデータのまとまりを整理してテーブルを分ける (社員 → 部門 など)	第二正規形
・分けたテーブルからさらにコードなどで特定できるデータのまとまりを整理してテーブルを分ける (社員 → 部門 → 事業部 など)	第三正規形

非正規化

案件によっては「テーブルを正規化しないほうが運用・開発しやすいので正規化しないでおく」という時があり、その場合「非正規化」と呼ばれることがあります。

だいたいは下記のようなケースで非正規化がされます。

- 一般的に正規化するとテーブルの結合などの処理が必要になることが多く、データをまとめて出力したいときなどに処理時間がものすごくかかってしまう懸念があるとき
  (膨大なデータの集計が必要な場合など)
- 正規化すると特別なケースだけ値を変えることがしづらくなるときがあるため、例えば「受注テーブルと製品マスタを分けたが、受注データに登録する製品名をこのときだけは元の「製品 A」から「製品 A (特注)」など手入力できるようにしたい」などの要望に応えられるようにするとき
- 「製品マスタの名前を変えたら過去の受注データの名前が変わってしまった」のような、過去のデータ(トランザクションデータ)が変わってしまうことを簡単に防ぎたい時
  ※マスタデータに適用開始・終了日時を持たせて防ぐアプローチもあります

ただ、非正規化すると複数のテーブルに似たようなデータが格納されることになり、注意深く設計・運用しないと整合性が崩れる場合があるのでどのようにするかは案件によって検討する必要があります。
(「One fact in one place」(1 つの事実は 1 つの場所に)、「二重管理は避けよ」、ということがよく言われます)

トランザクション (トランザクション処理)

トランザクション処理は、簡単に言うと「データベースに対する一連の操作をひとまとまりとして扱う処理」のことです。

データベースにデータを登録する時、複数のテーブルにデータを登録・更新・削除する必要がある場合がよくありますが、複数のテーブルに対する操作の際になにか 1 つの操作が失敗してシステムが止まると整合性が崩れてしまう場合があります。

それを防ぐために、複数の操作を 1 つの操作のまとまり(トランザクション)として扱って処理します。

トランザクション処理の流れ (コミット、ロールバック)

トランザクション処理では、下記のような流れで処理を進めていきます。

#	操作	SQL での指示
1	トランザクションの開始	BEGIN TRANSACTION
2	データ取得、登録、更新、削除などの処理	SELECT, INSERT, UPDATE, DELETE
3	コミット (トランザクション処理の確定)	COMMIT

2 の部分でエラーになった場合は、ロールバック (ROLLBACK) によって処理をキャンセルすることで、「中途半端なデータが残ってしまった」ということを防ぐことができます。

ロック

ロックは、データベースに対する操作中に別の場所からの操作が割り込みで入ってこないように一時的にデータを保護する仕組みです。ロックにはおおまかに「悲観的ロック」「楽観的ロック」の 2 つの種類があります。

悲観的ロック

「どんな操作が来てもいいようにあらかじめデータをロックして他からの操作が割り込むことをブロックしよう」という方式のことを「悲観的ロック」と呼びます。

主要な悲観的ロックには、下記のような種類があります。

行ロック	対象の行のみをロックする ・SELECT … FOR UPDATE (Oracle 等) ・SELECT … WITH (ROWLOCK,XLOCK) (SQL Server)
テーブルロック	対象のテーブル全体をロックする ・LOCK TABLE (Oracle, PostgreSQL) ・LOCK TABLES (MySQL) ・SELECT … WITH (TABLOCKX) (SQL Server)

悲観的ロックは、ロックのかける手順等によって多方面でロックの待ち合いになる「デッドロック」が起こる可能性があります。

楽観的ロック

「操作前に監視対象のデータを取得しておいて、操作時にデータが変わっていたら他から操作されているから処理をとりやめよう」という方式のことを「楽観的ロック」と呼びます。楽観的ロックは、下記のような流れになることが多いです。

1. テーブルに更新日時やバージョン列を用意しておき、更新したら値を変更するように開発者間で取り決めをしておく
2. 更新処理の直前に、対象行の更新日時(またはバージョン)を取得する
3. 更新処理に必要な処理などを行う
4. 更新の UPDATE を実行する際、対象行を特定する条件 + 2 で取得した更新日時(またはバージョン)を条件(WHERE)にする
   (ア) 他から更新されていた場合更新日時が変わっているため、WHERE の条件で行が抽出できず UPDATE 対象件数が 0 件になる → 失敗として扱う
   (イ) 他から更新されていなければ、UPDATE が成功する

照合順序 (COLLATE)

「照合順序」(COLLATE ※「これーと」, Collation)は、文字の順番(大きい、小さい、同じ)を示したものです。

→ 詳細は「照合順序(COLLATE)について」の資料を参照してください。