LibreOffice　Base マニュアル(入門編)　組込 DB：Firebird

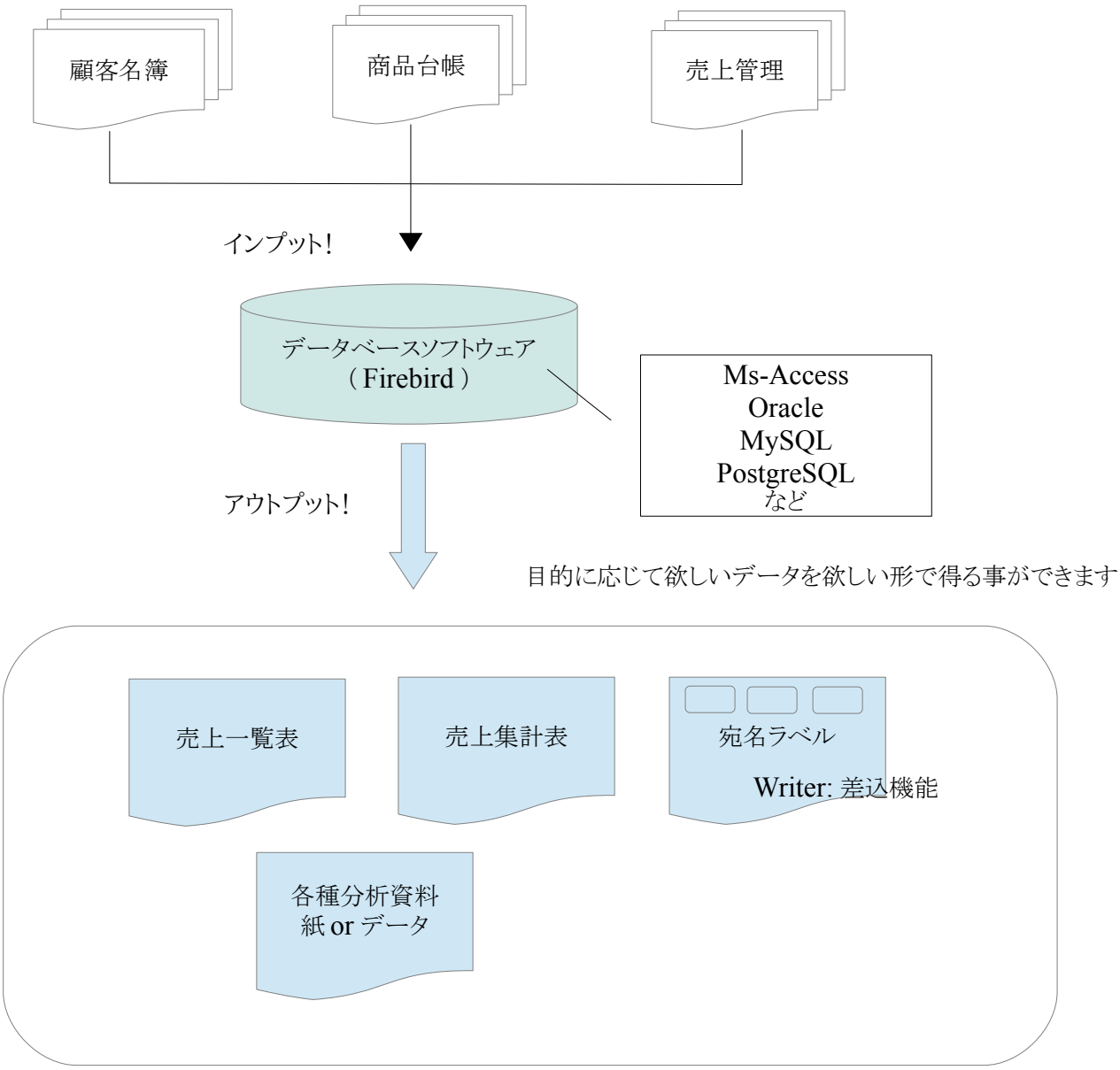
～目次～

下線部をクリックすると該当ページへ移動します、目次へ戻る場合は各ページのページ番号をクリックして下さい

データベースとは？

データベースとは、特定のテーマや目的に沿って集められたデータの集合体のことで、例えば「顧客名簿」「商品台帳」「売上管理台帳」などの帳票もデータベースと考えることが出来ます。

データベースを PC 上で管理するソフトウェアが「データベースソフト」です。
データベースソフトを利用することで、さまざまなデータベースを管理することが可能となり、欲しい情報を欲しい形で容易に取り出すことができるようになります。
表計算ソフトもデータの加工は可能ですが、表計算ソフトがデータ本体を操作するのに対し、データベースでは仮想表（クエリー）を用いてデータ本体に手を加えずに加工することができます。（ビューやストアドプロシージャを利用する場合もありますが、BASE では利用しませんので割愛します）



※欲しい形のデータをクエリーで作成し、そのクエリーの保存により、クエリーの実行だけで、欲しいときに欲しい形でのデータ出力を実現しています。

※Access と違い、LibreOffice-Base のレポート機能を使って宛名ラベルは作成できません。
宛名ラベルは Writer の差込印刷機能を使って作成します。（Writer から BASE 機能を利用して作成するため、BASE がインストールされている必要があります）

リレーショナルデータベース管理システム(RDBMS)とは?

リレーショナルデータベースとは、データをテーマや目的に合わせて細かく分類し、分類したデータに管理項目(フィールド)を設け、データ項目同士を識別コードによって関連(リレーション)付けした構造を持つデータベース管理方式で、データ管理方法では現在、最も多く利用されています。

【Relational DataBase Management System】　RDBMS と呼ばれる
リレーショナルデータベースデータ管理方式に基づいて設計されたデータベースのこと。
1970 年に IBM 社の Edgar F. Codd 氏によって提唱されたリレーショナルデータモデルを管理するソフトウェア。
リレーショナルデータベースとは、1 件のデータを複数の項目(フィールド)の集合として表現し、データの集合をテーブルと呼ばれる表で表現する方式です。
　ID 番号や名前などの識別キーとなるデータを利用することで、データの結合や抽出を容易に実施できます。
データベースとしては最も広く普及している方式で、データの入力ミスが少なく管理も容易になるだけでなく、重複したデータを登録しない仕組みのため、ディスク容量の節約などの利点があります。

RDBMS の特徴

1. 1 件のデータを複数の項目(フィールド)で構成された表(テーブル)で管理します
2. 登録したデータには番号やコードを付与し、一意(ユニーク)の値として登録します
3. データとデータ間を結びつける関連管理にはユニーク値を利用します(リレーショナル)
4. 大きく分けて「固定情報(氏名等)を登録するマスターデータ」とマスターデータ同士を組み合わせて発生する取引を管理する「取引データ・明細データ」の2つがあります
5. マスターデータと取引データとの関係は基本的に 1 対多(1:n)の関係を示します

※一意:「重複が許されない」という考え方、ある値を指定することでデータを特定できることが要件
(例:商品マスタ管理で利用する場合、商品を特定するコードを一意に設定する必要があります)
一意のコードを設定し識別コードとすることを**「プライマリーキーを設定する」**と呼びます

RDBMS で利用するオブジェクト(Ms-Access/LibreOffice_BASE の場合)

1	データを保存するオブジェクト	・・・・・・・	テーブル(一部ビューも作成可)
2	データを加工するオブジェクト	・・・・・・・	クエリー
3	入力・表示用オブジェクト	・・・・・・・	フォーム
4	印刷レイアウトのオブジェクト	・・・・・・・	レポート

※Oracle や SQLserver・PostgresSQL などはテーブルとクエリー(SQL)を利用し、データの入出力にはフロントエンドとして HTML や ASP や PHP などの WEB 画面を利用したり、Access や BASE などのパッケージデータベースソフトを利用します。

【主なリレーショナルデータベースソフト】・・・Access と BASE 以外は埋込ではなくサーバーでの運用が主です

1. Oracle　・・・米オラクル社の商用製品
2. SQLserver・・・米マイクロソフト社の商用製品
3. MySQL・・・米オラクル社が著作権を保持している OSS(オープンソース)
4. PostgreSQL・・・PostgreSQL Global Development Group が開発している OSS(オープンソース)
5. Ms-Access・・・米マイクロソフト社の商用製品(主にパーソナルユース)
6. LibreOffice-BASE・・・The Document Foundation(TDF)が開発している OSS(オープンソース)

【業務用システムのデータベース構成例(概要)】
入出力・・・・PowerCOBOL/JAVA/ASP/PHP/HTML/Excel
RDBMS・・・Oracle・PostgreSQL・MySQL

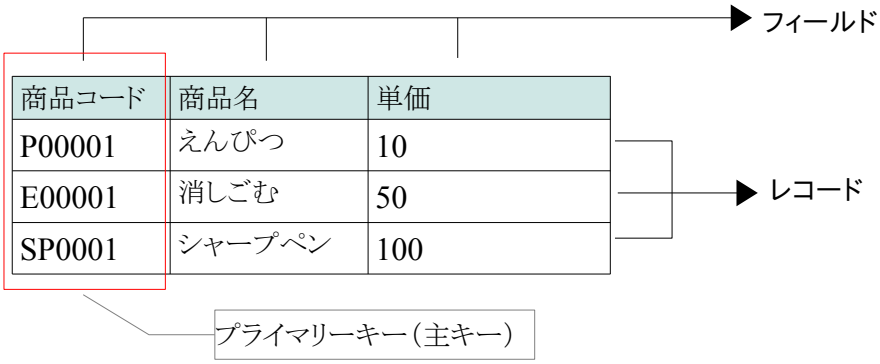
【BASE では組込データベースを選択可能】
・埋込 (Embedded)DataBase(DB)・・・LibreOffice-BASE の ODB ファイルに同梱可能なデータベース(DB)
　　DB 埋込によって Ms-Access のようなファイル運用が可能となりました。
・HSQLDB と Firebird とは?・・・BASE ファイルに埋込可能なデータベース本体
・HyperSQL-DataBase・・・HSQLDB:Java100%の DB、SQL の実行での制限が多い
・Firebird-旧 Borland の InterBASE が土台(LibreOfficeVer4.2 から実装・HSQLDB と選択可能に)
※LibreOffice5.4 から Firebird-Ver3 が採用され、SQL-92 準拠+SQL-99 の一部取入れや標準関数の増加を含め運用しやすくなっています。

フィールドとレコード

データを格納する役割を担うテーブルには、格納するデータ別に項目（フィールド）があり、1つ1つのデータをレコードという単位で管理しています。

RDBMS（リレーショナルデータベース）では各データを縦横の表で表現します。

表計算ソフトで言う「カラム(Column):列」がフィールド・「ロウ(Row):行」がレコードになります



各フィールドには1つの意味を持つ値だけが格納されます。
例えば：単価フィールドに入るのは単価だけであって商品名や商品コードが入ることはありません（フィールド名には格納されるデータをイメージできる名前を設定しましょう）

重複しないコード（プライマリーキー）：上記例では商品コードをプライマリーキー（主キー）として設定しています
商品コードはプライマリーキーですから、重複しない値を設定しなければなりません。
また、主キーに既存のコードを利用する場合は、現在の重複確認だけでなく、過去に重複がなかったか？また、今後も重複が発生しないことを確認する必要があります。

※データベース特有の堅牢さが問題になる
データベース運用は表計算ソフトと違い、基本的なデータ構造に漏れや修正が発生した場合、追加・修正ではなく基本構造から作り直しになることがあります。
データの型やテーブルの分割などからデータベース設計に入る際にも、データ構造の確認と重複の可能性については、単なるシステム運用上だけでなく、将来的な業務変化（店舗統合や店舗廃止）時に対応できることを前提に検討しておく必要があります。

【表計算ソフト　Calc や Excel との比較イメージ】
表計算ソフトは思い付きで作り始めても何となくできていくという感じ。（項目の追加も修正も簡単）
BASE などのデータベースソフトでは、作り始める前に、どんな出来上がりの形が必要が、どんなデータをどんな形（テーブル）で持って、どう繋ぐ（リレーションする）かを十分考えてから作る必要があります。
テーブルの項目やテーブルの持ち方などにも留意しておかなければなりません。

【テーブル項目の追加がある場合、最後に追加されます（割り込みはできません）】
Firebird を含むデータベースでは、基本的に一度テーブルを設計し保存した後、既存項目と項目の間に新規項目を割り込み追加することはできません。（Access を除く）
追加する項目はテーブルの項目間ではなく、最後に追加されます。
追加修正が発生した場合、フォームのタブストップを調整することで、フォームに入力する順番を前後させることが可能です。（見かけ上ですが入力順を変更できます）

テーブルとデータ型

データを格納する役割を担うテーブルには、格納するデータ別に項目を設定し、さらに項目毎にデータ型を定義しなければなりません。(必須事項)

【データ型】
格納するデータ内容によってデータ型を指定し、一つのデータ項目には一つの型を持つデータが格納されるようにします。
売上日を登録する項目には売上日付といったように、データ内容が推測できるような項目名(フィールド名)をつけると後の運用が楽になります。
売上日項目に入荷日や出荷日が登録されるなど、1 つの項目に複数種別のデータが登録されることはありません、したがって管理する実際のデータ項目の増加に比例してテーブルのデータ項目も増加します。

【Access データ型と Base(Firebird)データ型との比較】

Access のデータ型	BASE のフィールドタイプ(データ型) Firebird 3.0 参照
テキスト型	可変テキスト(VARCHAR)　固定テキスト(CHAR)
メモ型	可変テキスト(VARCHAR)
バイト型	最短整数(SMALLINT)
整数型	整数(INTEGER)　符号付 32 ビット整数
長整数型	整数(INTEGER)　符号付 32 ビット整数 長整数(BIGINT)　　符号付 64 ビット整数 Dialect3/FB1.5 以降で使用可能
単精度浮動小数点型	IEEE 単精度実数(FLOAT)
倍精度浮動小数点型	倍精度浮動小数点(DOUBLE PRECISION)
日付/時刻型	日付(DATE)/時刻(TIME) 日付時刻の時は(TIMESTAMP)
通貨型	NUMERIC または DECIMAL　長整数(BIGINT)
オートナンバー型	整数(INTEGER) 長整数(BIGINT) NUMERIC または DECIMAL だとエラー
YES/NO 型	はい/いいえ(BOOLEAN):True/False/Unknown
OLE オブジェクト型	なし
ハイパーリンク	なし

※テキスト型の長さは1文字 4 バイトとし、VARCHAR では(バイト数による超過分切捨て)・CHAR では(文字数とバイト数)でチェックするようです。
※INTEGER については Basic 変数で使う Integer の範囲と混同しないように注意します
　Basic マクロ変数での Integer 範囲:-32768～32767 の整数
　データ型 INTEGER 範囲:-2147483648 ～ 2147483647 の整数
　INTEGER 型では不足する場合に BIGINT 型を利用する運用を行います
※BASE の場合、埋込 Firebird 以外へ接続した場合は接続先データベースに呼応したデータ型が表示されるようです

※数値型は Access の場合、お金は通貨型で OK でしたが、Firebird の場合ハッキリとした型がわかりません。小数点が出るか出ないかが重要らしく
・整数:19 桁までは BIGINT, それより大きい場合は NUMERIC を利用
・お金関連の項目(フィールド)は NUMERIC または DECIMAL を利用(精度を保つため)
といった対応方法があるようです。型設定の参考にしてください。

データベースを設計する

データベースを構築する時、まず最初にしなければならないのは「データの目的や用途を明確にする」ことです。データを利用する目的や用途がハッキリしていなければ、集積データとして何をどのような形で集めれば良いかが不明瞭となり、活用が難しいデータベースとなってしまいます。

1. 利用目的を決めましょう
   まず最初にデータベースの利用目的を決めます。
   いろいろなデータを管理したい気持ちが出ますが、最初は欲張らずに利用目的を絞り込み、データの用途をハッキリさせることが必要です。

2. 印刷結果や入力画面のイメージを考えます
   目的が決まったら、次にどのような画面で入力するのか?どのような形で印刷するのか?をイメージし、手書きで良いので簡単なレイアウト表などを作成してみます。
   この際、現在利用している帳票や入力している表計算シートを参考にしてみると、イメージしやすくなります。
   一覧表では日付順に並んでいた方が良いですか?　個人毎の明細は必要ですか?　あいうえお順に並べる予定はありませんか?　イメージを膨らませることでデータベースを利用した作業を具体的に把握することができます(ここでのイメージが次のテーブル設計で役に立ちますのでしっかりイメージしましょう)。

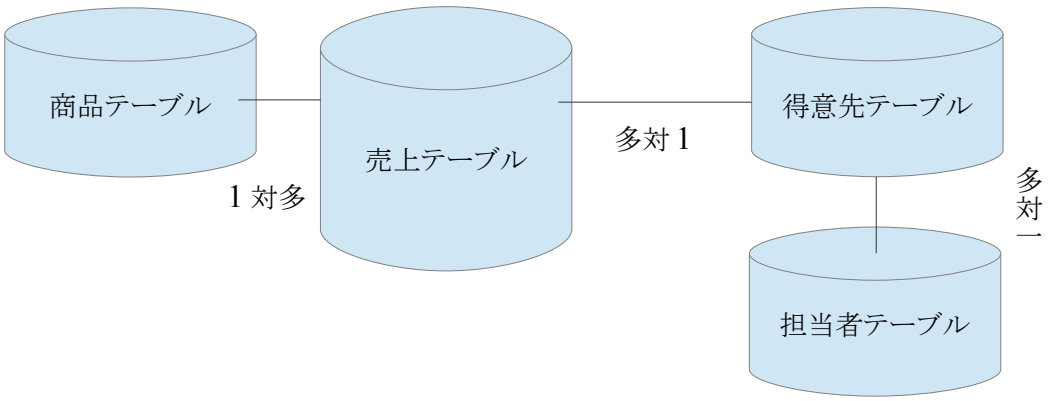
3. テーブルを設計 1(必要な項目を特定します)
   イメージした帳票や入力画面を元に、テーブルを設計に必要なデータ項目を列挙します。
   必要な情報に必要なデータ項目を漏れなくテーブル項目としなければなりません。
   - 宛名シール用データなら郵便番号が必要ではありませんか?
   - あいうえお順に並び替える場合は「ふりがな」項目が必要です
   - 商品単価に仕入価格と販売価格が必要ではありませんか?
   - 通知書送付用に敬称(様・御中)が必要ではありませんか?
   - 消費税額を別途管理する必要はありませんか?

4. テーブルを設計 2(データ属性を基本にデータを切り出します)
   イメージした帳票や入力画面を元に特定したテーブル項目の属性を見極め、属性毎にテーブルとして切り出す作業を行います。
   (例)
   - 商品コードは商品の属性です
   - 商品名は商品の属性です
   - 得意先は顧客の属性です
   - 担当者コードは担当者の属性です
   - 売上日は売上の属性です

   属性を1テーブル、また区分した項目を 1 フィールドとして切り出すことができます(1 属性 1 テーブル)。

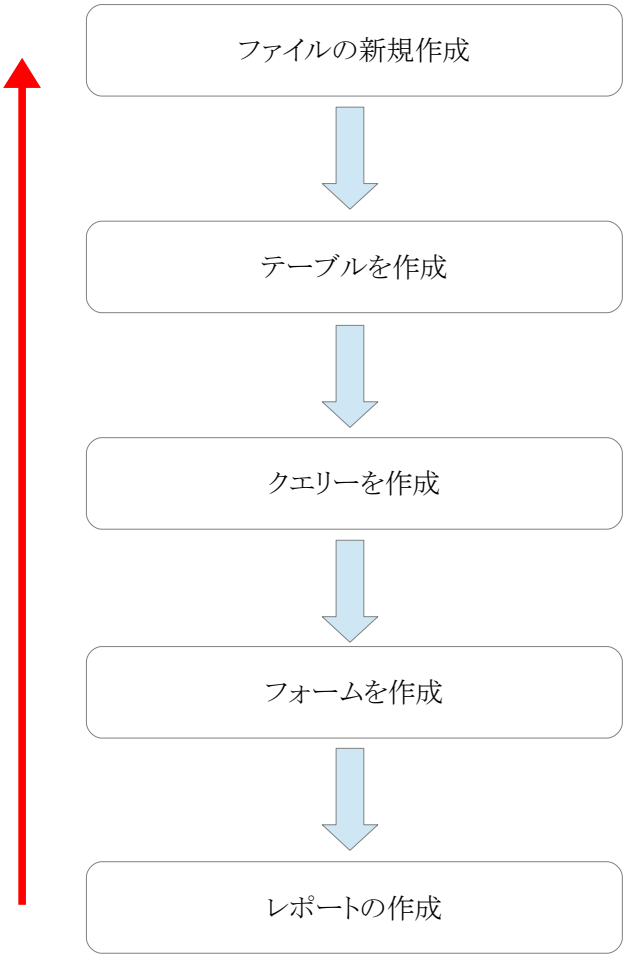
5. テーブルを設計 3(テーブルの関連を考えます)
   設計したテーブル同士は、どの項目で関連付けたら元の帳票に戻るか?をイメージしてテーブル間の関連「リレーション」を考えます。
   (例)
   1. 売上テーブルに商品テーブルが商品コードで関連しています(売上に商品を反映できる)
   2. 売上テーブルに得意先テーブルが得意先コードで関連しています(売上に得意先を反映できる)
   3. 得意先テーブルに担当者テーブルが担当者コードで関連しています(得意先の担当者が判ります)

テーブル同士の関連を図で表示してみると



※ 1人の担当者が複数の得意先を担当している

6. データベース構築の流れ(データから帳票へ)



odb ファイルにはテーブル等の各オブジェクトを格納する入れ物としての役割があります。

データを格納するための表を作成します。

テーブルを結合した仮想表を作成します。
条件に合うデータを抽出・計算・集計します。

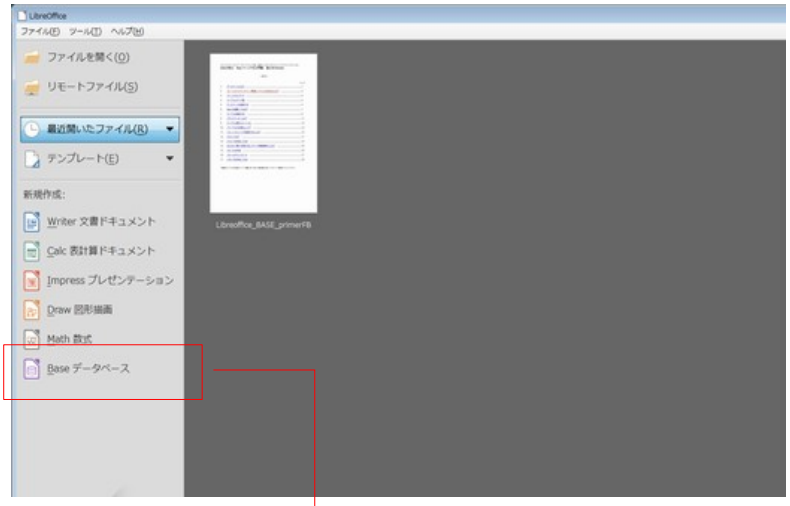
データ入力時の画面を作成します。

データを印刷する帳票を作成します。
必要に応じてデータの並べ替えや必要な形式へ変更した印刷帳票も作成します

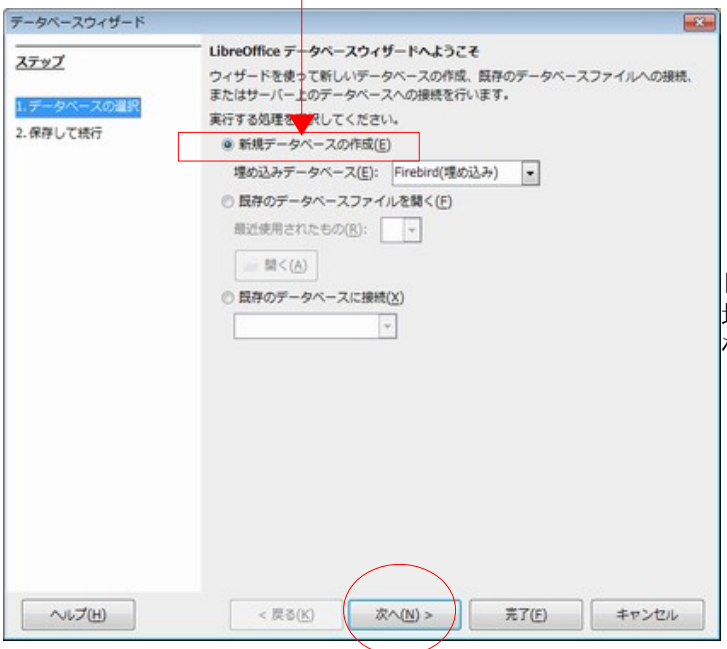
データ設計する際の考え方
(帳票から必要なデータを考える)

Base を起動してみます

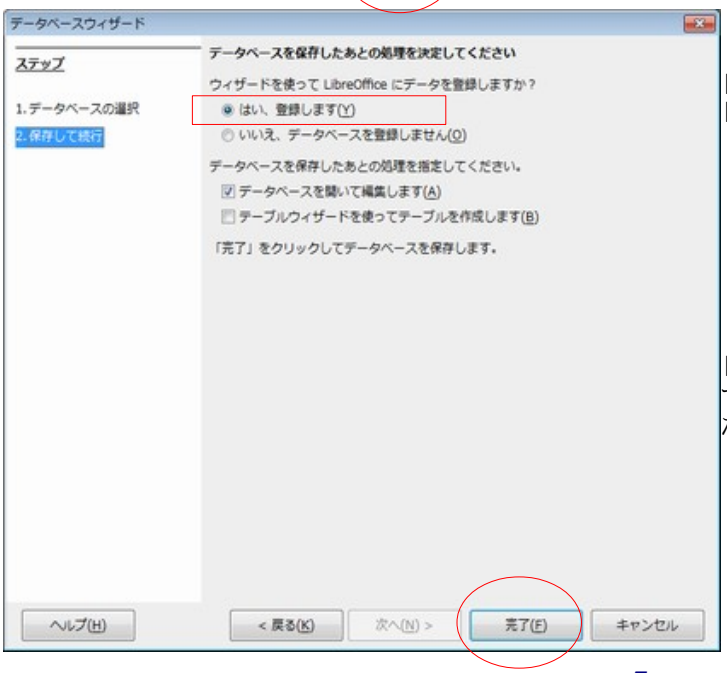
LibreOffice を起動します



データベースをクリックします



「新規データベースの作成」を選択し
埋め込みデータベースが Firebird（埋め込み）に
なっているのを確認し「次へ」をクリックします

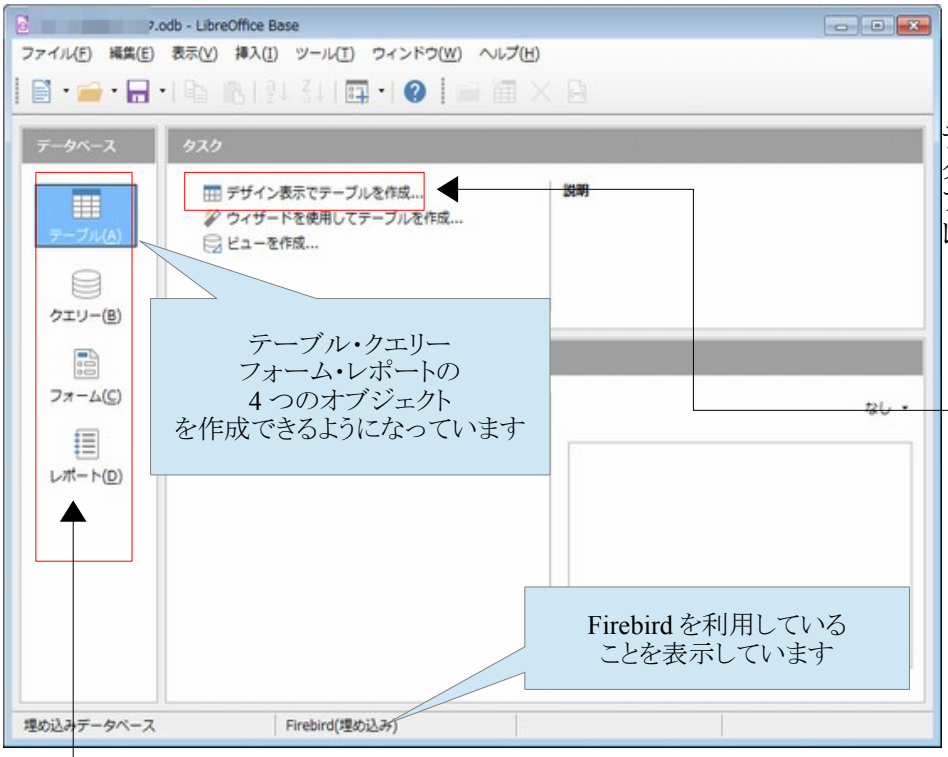


「はい、登録します」を選択し
「完了」をクリックします

「名前を付けて保存」のダイアログが表示されますので適当な名前を付けてファイルを保存してください。

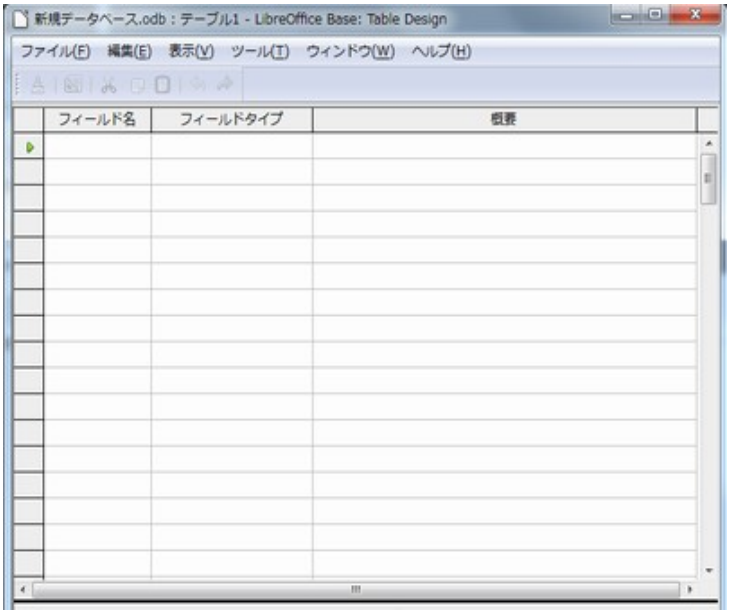
LibreOffice_BASE　起動画面

BASE の基本画面です



テーブル　:データの入れ物
クエリー　:仮想表
フォーム　:入力用画面
レポート　:帳票

テーブルを設計しますので　　枠内のオブジェクトバーから≪テーブル（A）≫を選び、右横のタスクから「デザイン表示でテーブルを作成」を選択します



フィールド名:項目名

フィールドタイプ:データ型

それぞれのフィールド毎に設定していきます

概要には留意点等を記入しておきます

テーブルを設計する

テーブルはデータを格納する為に作成するオブジェクトのことで、データ保管庫といったイメージでかまいません。項目（フィールド）を決めたら、項目内容に合ったデータ型を指定します。データ型は利用するデータベースによって細かな違いがありますが、大きく「数値」「テキスト」「日付」「はい・いいえ」に分類することができます。

【データ型】　BASE（Firebird）のデータ型（◎や○のところを押さえておくだけでも良い）

分類	データ型名	範囲
文字列◎	CHAR(m) VARCHAR(m)	半角英数（ASCII）　1～32767 文字 全角（SJIS_0208）　1～16383 文字 CHAR(m)は m 文字の固定長 VARCHAR(m)は最大 m 文字の可変長
整数◎	SMALLINT INTEGER	-32,768 ～ 32,767 -2,147,483,648 ～2,147,483,647
長整数◎	BIGINT	符号付 64 ビット整数　Dialect3/FB1.5 以降で使用可能
実数	FLOAT DOUBLE PRECISION	3.4×10 の-38 乗～3.4×10 の 38 乗　有効桁数 7 桁 1.7×10 の-308 乗～1.7×10 の 308 乗　有効桁数 15 桁
固定小数点	NUMERIC（m,n） DECIMAL（m,n）	有効桁数 m = 1 ～ 15 小数点以下の桁数 n=1～15（m >= n） 最高 m 桁の有効数字を格納 有効桁数 m = 1 ～ 15 小数点以下の桁数 n=1～15（m >= n） 最低 m 桁の有効数字を格納
日付◎	DATE TIME TIMESTAMP	西暦 100 年 1 月 1 日～32768 年 2 月 29 日までの年月日 00:00:00～23:59:59 までの時分秒 西暦 100 年 1 月 1 日～32768 年 2 月 29 日までの年月日と時分秒
その他	BLOB	グラフィック，文字などの大量データに使用:可変長
論理型○	BOOLEAN	真理値の「真 = true」と「偽 = false」

データベースでは項目の事をフィールドと呼びます。
フィールド毎にデータ型（フィールドタイプ）を定義することで規定以外の値が保存されることを防止します。

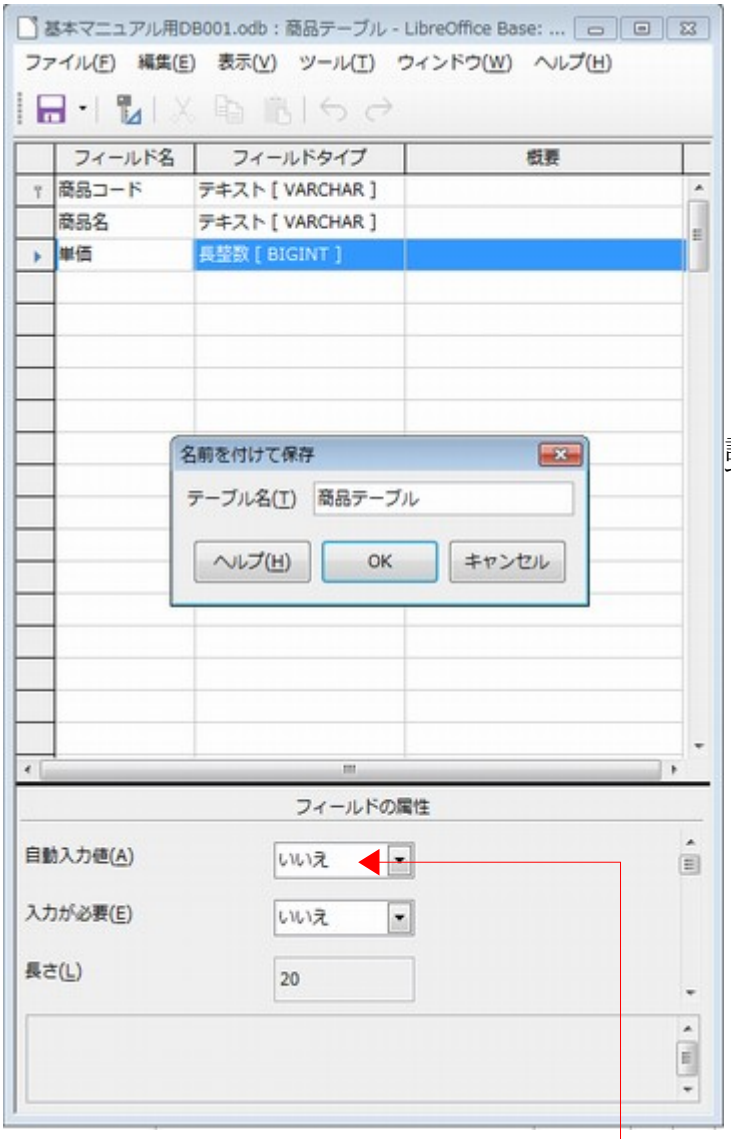
【データ型の目安】
テキスト型・・・文字・計算しない数字（郵便番号・電話番号・各種コード等）、長文には VARCHAR を利用します
数値型・・・金額や整数・少数点を含む値に利用します（ほとんど INTEGER と BIGINT で事足ります）

【データ長】
データにはデータ型の他にデータの長さを指定する場合があります。
マルチバイト文字を利用する時は文字数の制限と文字のバイト数の制限が出ますので注意してください。
テキスト型の1文字の長さは 4 バイトとしてカウントされるようです。
・VARCHAR は（バイト数による超過部分切捨て）　VARCHAR 型で長さ1の場合 4 バイト文字 1 文字まで
・CHAR では（バイト数だけでなく文字数）でもチェックするようです。

【テーブルの設計例】
商品テーブル

商品コード	商品名	単価
P00001	えんぴつ	10
E00001	消しごむ	50
SP0001	シャープペン	100

商品テーブルの設計では以下のように設定します



商品コード：プライマリーキー

設定終了後「名前を付けて保存」
で適当な名前を付けて保存してください。

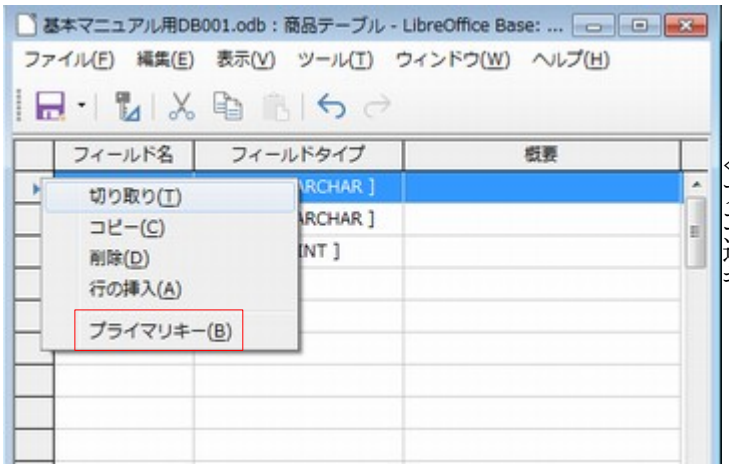
≪Point≫
ココを「はい」にすると自動採番されるように
なります（INTEGER と BIGINT のみ）
Access のオートナンバー型と同じ

長さ(L)の目安

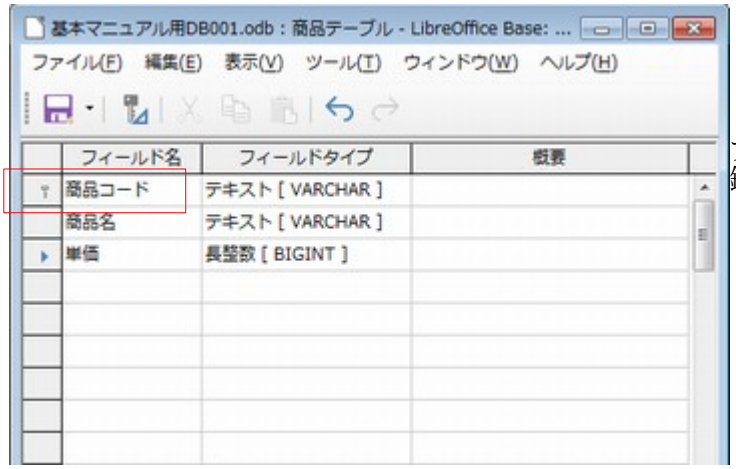
- 数字を使ったコードなら最大桁数(4 ケタの商品コードなら 4)
- 商品名などの文字データ(商品名に利用されてそうな文字数、あっても 50 文字以下だろう!なら 50)
- 金額などのデータ(型の既定値を設定：整数型なら 10　長整数型なら 19)

プライマリーキーとは?

プライマリーキーとはテーブルにデータを格納する際に、重複しない固有のデータ(一意のデータ)として識別する為に設定するコードです。(Access では「主キー」と呼ばれています)
プライマリーキーを設定することにより、登録されたデータ(レコード)は、それぞれ固有のデータとして認識されデータの検索や集計が行えるようになります。
プライマリーキーは「商品コード」「顧客 ID」など重複が発生しないコードや番号のフィールドに設定するようにしましょう。



≪設定方法≫
プライマリーキーを設定したいフィールド(商品コード)横の□を右クリックし、商品コードを全て選択状態にして(青地になります)プライマリーキーを選択します



プライマリーキーがセットされると項目名の左に鍵マークが表示されます

※プライマリーキーの自動採番について
売上管理する「売上テーブル」などでも売上 1 件毎にプライマリーキーが必要です。
売上テーブルなどの契約情報を登録するテーブルでは、入力されるコード類をプライマリーキーに設定せず、自動採番する連番フィールドをプライマリーキーにしたい場合があります。
その場合は「売上管理 ID」欄などのフィールドを設定し、フィールドタイプを「整数 Integer」や「長整数 Bigint」に設定した後、自動入力値(A)を「はい」にすることで自動的に連番をセットすることができます。
Access のオートナンバーと同じ機能になります(前ページ≪Point≫参照)。

【複数のフィールドをプライマリーキーにするには?】
通常プライマリーキーは1フィールドを指定しますが、複数のフィールドを同時指定することで設定する場合もあります。(例:都道府県コード+店舗管理番号)
Ctrl キーを押したまま、設定予定の各フィールドを選択すると複数のフィールドを選択することができ、複数のフィールドにプライマリーキーを設定することができます。

テーブル分散とリレーション

リレーショナルデータベースはテーブルという単位でデータを分散保持し、分散しているデータをリレーションと呼ばれる関係で構築することによって、必要なデータを再構成する仕組みであることは理解できました。
ここで一つの疑問が出てきます、利用するデータの形が決まっているのであれば、1つのテーブルに全てを保存していたほうがスッキリと判りやすくて良いのではないか?ということです。
なぜ、わざわざテーブルを分散させ、必要な時にはクエリーによってデータを再構成して利用するという面倒なことをおこなっているのでしょうか?。

分散させて管理している理由はいくつかありますが、代表的な事項として以下の3つが挙げられます。

1. データの保守が簡単・確実に行える
2. データ容量を少なくすることができる
3. 必要なデータだけを取り出すことが容易にできる

【データ例】　一覧表で管理した場合

購入日	商品名	数量	得意先	住所	電話番号
2013/04/15	えんぴつ	10	得意先 A	福岡市南区××	092-111-1234
2013/04/15	えんぴつ	5	得意先 B	福岡市東区△××	092-111-2222
2013/06/15	消しゴム	15	得意先 D	福岡市西区××□	092-111-3333
2013/06/15	消しゴム	2	得意先 A	福岡市南区××	092-111-1234
2013/05/15	シャープペン	2	得意先 F	福岡市早良区××	092-111-4444

【データ例】　分散テーブルで管理した場合

売上テーブル

売上No.	購入日	商品コード	数量	得意先コード
1	2013/04/15	P00001	10	0001
2	2013/04/15	P00001	5	0002
3	2013/06/15	E00001	15	0004
4	2013/06/15	E00001	2	0001
5	2013/05/15	SP0001	2	0006

商品テーブル

商品コード	商品名	単価
P00001	えんぴつ	10
E00001	消しごむ	50
SP0001	シャープペン	100

得意先テーブル

得意先コード	得意先名	住所
0001	得意先 A	福岡市南区××
0002	得意先 B	福岡市東区△××
0003	得意先 C	福岡市西区××□
0004	得意先 D	福岡市南区××○○○
0005	得意先 E	福岡市早良区 123××
0006	得意先 F	福岡市早良区××

売上テーブル上ではコードで登録されている

商品名が変更になっても、得意先名が変わっても、住所が変更になったとしてもマスターになっている各テーブルのデータを修正するだけで反映されます。

上記例では、得意先の名称変更発生時でも得意先テーブルの得意先名を書き換えるだけでリレーションされた各テーブルの値も書き換えた状態になります。

「テーブルの正規化」って?

テーブルの正規化と聞くと何か特別な作業を行うイメージがありますが、実際はデータベースをより効率的なものにするためにテーブルの設計を考え直す作業のことです。
テーブルの正規化では、前項で行ったようにテーブルを分割し分散管理する作業が中心となります。
正規化には第一から第三の段階(理論上は第五までありますが非実用的なので無視して良い)があり、それぞれ「効果」が異なります。
レコード更新がほとんどないテーブルや、データベースシステムの性能・動作させる PC の能力など、諸々の都合によって、あえて正規化しない場合もあります。
※単なるデータ加工(集計や演算)を行う場合は、「テーブルの正規化」を意識する必要はないと考えます。

≪第一正規化≫
テーブルの項目に複数の値が入らない状態にすることを「第一正規化」といいます。
1 行 1 項目 1 データにした後、計算して求められる「合計額」「消費税額」などがある場合は除外します

非正規化状態(ID はユニーク値:プライマリーキー)　　販売品が 2 列ある

ID	得意先	業種番号	業種	販売品	販売品
1	得意先 A	01	製造	AAA	BBB
2	得意先 B	02	小売	FFF	VVV
3	得意先 C	02	小売	BBB	KKK
4	得意先 D	03	卸売	CCC	

販売品が複数列存在しています
(Excel などでお買い上げ明細等を管理すると、上記の形式になっている場合が多い)

第一正規化(1 行 1 項目に 1 データ)　販売品列が一つになりました

ID	得意先	業種番号	業種	販売品
1	得意先 A	01	製造	AAA
1	得意先 A	01	製造	BBB
2	得意先 B	02	小売	FFF
2	得意先 B	02	小売	VVV
3	得意先 C	02	小売	BBB
3	得意先 C	02	小売	KKK
4	得意先 D	03	卸売	CCC

販売品列を 1 つにすることで複数項目にデータが分散している状態を解消した結果、今度は ID が重複してしまうようになりました。

ID の重複を解消するため
第二正規化を行います

≪第二正規化≫（プライマリーキーで分割）
プライマリーキーが一意（ユニーク）になるようにテーブルを分割します
次に、プライマリーキーに従属した項目を切り離して別テーブルにします

ID	得意先	業種番号	業種
1	得意先 A	01	製造
2	得意先 B	02	小売
3	得意先 C	02	小売
4	得意先 D	03	卸売

ID	販売商品
1	AAA
1	BBB
2	FFF
2	VVV
3	BBB
3	KKK
4	CCC

≪第三正規化≫（プライマリーキー分割後のデータをさらに別コードで分割）→例では業種番号が別コード
「第二正規化」まで実施済のテーブルに対し、さらに分離できる項目を見つけ、それを独立したテーブルに分割し、キー項目以外の項目が互いに依存関係を持たないようにすることを「第三正規化」といいます。
難しい感じがしますが、要するに分割できるものは分割してしまおう、ということだと考えて大丈夫です。

得意先表

ID	得意先	業種番号
1	得意先 A	01
2	得意先 B	02
3	得意先 C	02
4	得意先 D	03

業種表

業種番号	業種名
01	製造
02	小売
03	卸売

販売表

ID	販売商品
1	AAA
1	BBB
2	FFF
2	VVV
3	BBB
3	KKK
4	CCC

実取引を想像してデータ構成を検討すると、すんなり第三正規化まで完了したテーブル設計ができるようです。

分散テーブルのところで解説しているように、データの属性を見極め、マスターテーブル（重複が認められないもの）と累積テーブル（取引情報などを登録する重複アリ）との違いを意識すれば良いと思います。

「リレーションシップを設定する」とは?

各テーブル間で共通しているフィールド(項目)を互いに関連付けることを「リレーションシップを設定する」と言います。
リレーションシップが設定された複数のテーブルを結合すると、あたかも一つのテーブルであるかのようにデータを取り扱うことができます。

プライマリーキーと外部キー
2 つのテーブル間にリレーションシップを設定するには、2 つのテーブルに共通のフィールドが必要です。
共通フィールドのうち「プライマリーキー」側のフィールドに対して、もう一つのフィールドを「外部キー」と呼びます。
「プライマリーキー」があるテーブルを「主テーブル」、もう一方の関連付けするテーブルを「外部テーブル」または「リレーションテーブル」と呼びます。

売上テーブル(リレーション T)

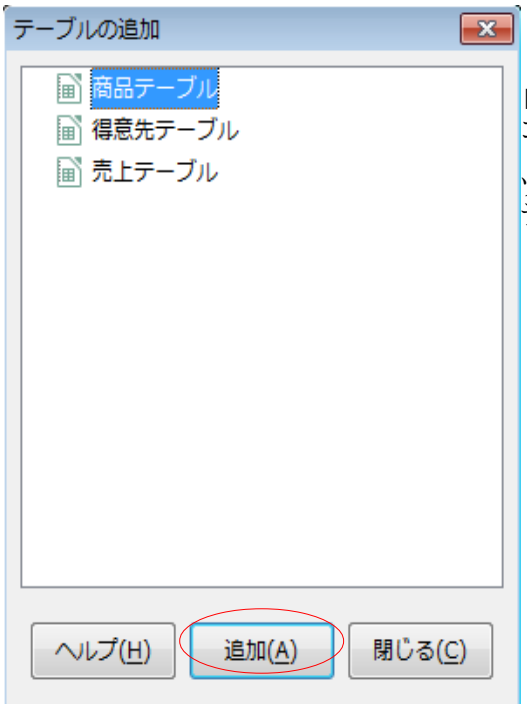
売上No.	売上日	商品コード	数量	得意先コード
1	2013/04/15	P00001	10	0001
2	2013/04/15	P00001	5	0002
3	2013/06/15	E00001	15	0004
4	2013/06/15	E00001	2	0001
5	2013/05/15	SP0001	2	0006

商品テーブル(主テーブル)

商品コード	商品名	単価
P00001	えんぴつ	10
E00001	消しごむ	50
SP0001	シャープペン	100

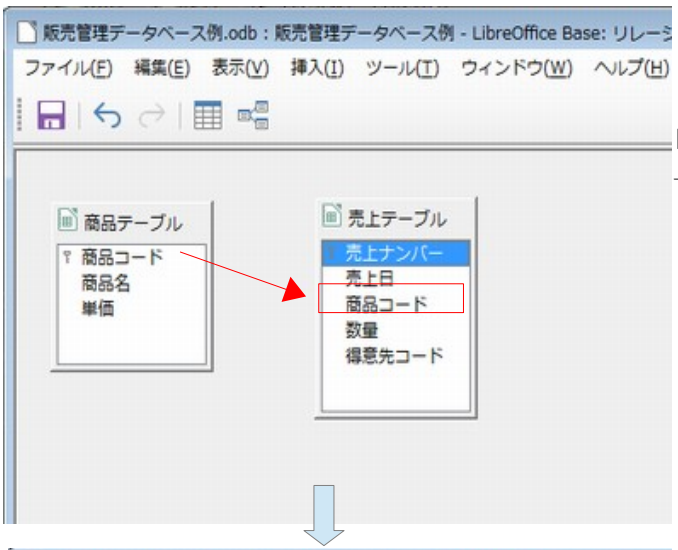
得意先テーブル(主テーブル)

得意先コード	得意先名	住所
0001	得意先 A	福岡市南区××
0002	得意先 B	福岡市東区△××
0003	得意先 C	福岡市西区××□
0004	得意先 D	福岡市南区××○○○
0005	得意先 E	福岡市早良区 123××
0006	得意先 F	福岡市早良区××

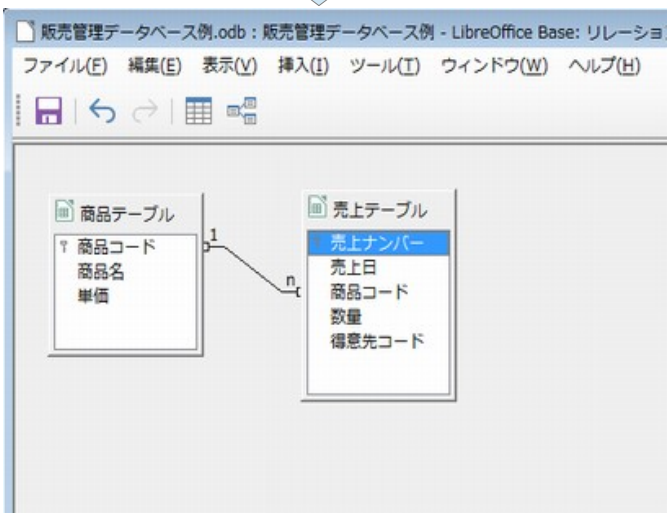


「商品テーブル」の≪商品コード≫と「売上テーブル」の≪商品コード≫にリレーションを設定します

ツール　→　リレーションシップ
テーブルを選択し「追加(A)」ボタンをクリックします



「商品テーブル」の≪商品コード≫を選択した後、「売上テーブル」の≪商品コード≫に向けてドラッグします。



1 対n （1:多）のリレーションが設定されました

1 対 1 のリレーションになる場合はテーブル分割が不要（どちらかのテーブルの項目として設定したほうが良い）の場合が多いです。

リレーションの編集
設定されたリレーションシップの動作を編集するとテーブル間のデータ管理の整合性を保つように設定できます。
（必ず設定しなければならないというものではありません）
例えば、「商品テーブルに無い商品コードを売上テーブルに入力してしまう」といったデータ矛盾を防止できます。

リレーションの結合線を右クリック → 編集

データ更新時の動作を指定する「更新オプション」
データ削除時の動作を指定する「削除オプション」

既定値:何もしない
カスケードの更新・削除:主テーブルのデータ変更に連動し外部テーブルのデータも更新される
null に指定:外部テーブルに NULL 値がセットされます
デフォルトの設定:テーブル既定値が反映される?
（NULL がセットされるようです）

クエリーとは？

クエリーとは、各テーブルに格納されたデータをいろいろな視点から加工するためのオブジェクトで、フィールドやレコードを欲しい形に加工したり、計算式を実行した仮想的な一覧表を作成できる機能です。
また、作成したクエリーを保存することもできるため、保存したクエリーを実行するだけで条件に合致したデータを取得することができます。

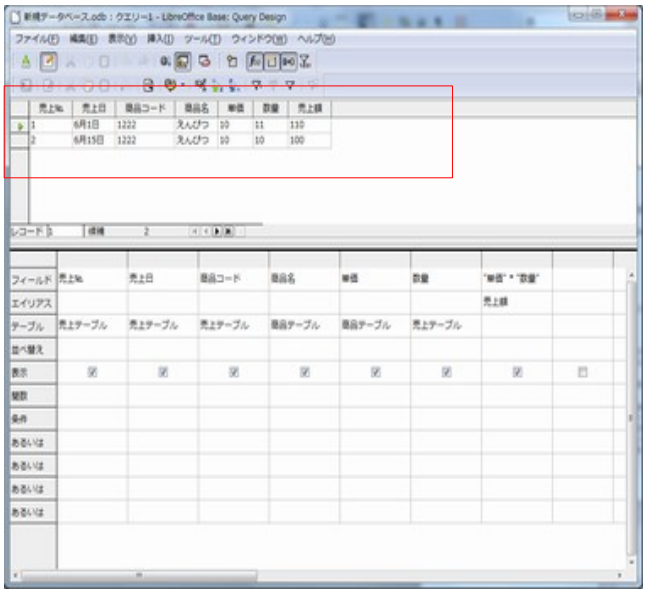
【クエリーでできること】

1. 必要なフィールドを抽出して組み合わせ、仮想的な一覧表を作成できます
2. 複数のテーブルを結合し、仮想的な一覧表を作成できます
3. フィールドのデータを基礎に計算することができます（単価×数量＝売上額など）
4. 抽出条件を設定しデータを抽出できます
5. データをグループ化して集計することができます
6. データを並べ替えることができます
7. データやテーブルを削除したり作成したりできます

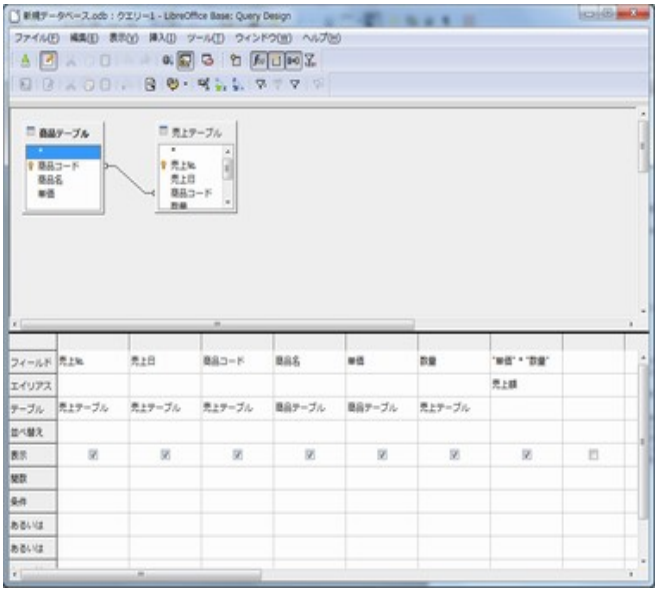
※Base のクエリーでは Access と違い、マクロ（OpenBasic/VBA）で作成したユーザー定義関数（Function 関数）を使えません
したがって Access よりも SQL 上で行う分岐や計算が非常に多くなってしまいます。

【クエリー＝SQL?】
Base だけに限らず、クエリーとは SQL 文（構造化問い合わせ言語）のことです。
データベースソフト（Base や Access）にはクエリーという機能があり、データベースに対して問い合わせ(検索、追加、更新、削除など)を行う際に利用します。
Base や Access などのデータベースアプリケーションでは、視覚的に作成したクエリーを保存しておいて、何度でも実行することができます。
クエリーをデザインすると、裏では SQL(Structured Query Language)文が自動的に作成されています、このことはホームページ作成ソフトが画面の裏で HTML 文を記述しているのに似ています。
クエリーから SQL とは逆に SQL 文でクエリーをデザインすることも当然可能です。
このクエリーと SQL という二つの関係は非常に密接で切り離せないものとなっています。
単純な問い合わせや集計はクエリーを使って作成できますが、副問い合わせやクエリー同士を連結するユニオンクエリーなどは SQL 文を直接記述する必要があります。

クエリー実行画面



デザインビュー画面

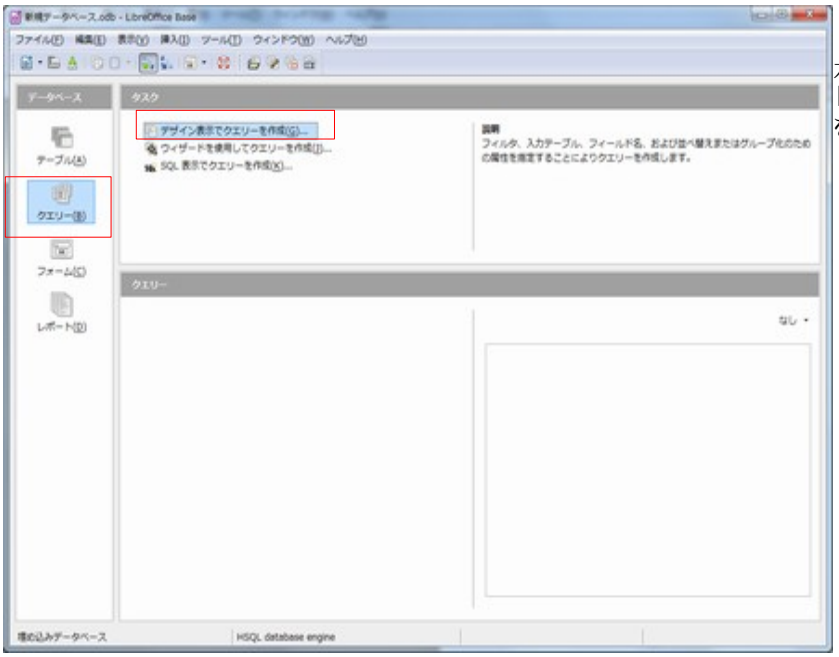


クエリーを作成してみる

クエリーを作成する方法は「デザイン表示で利用する方法」「ウィザードを使う方法」「SQL 文を記述する方法」の 3 つがあります。今回はデザイン表示を利用してクエリーを作成してみます。

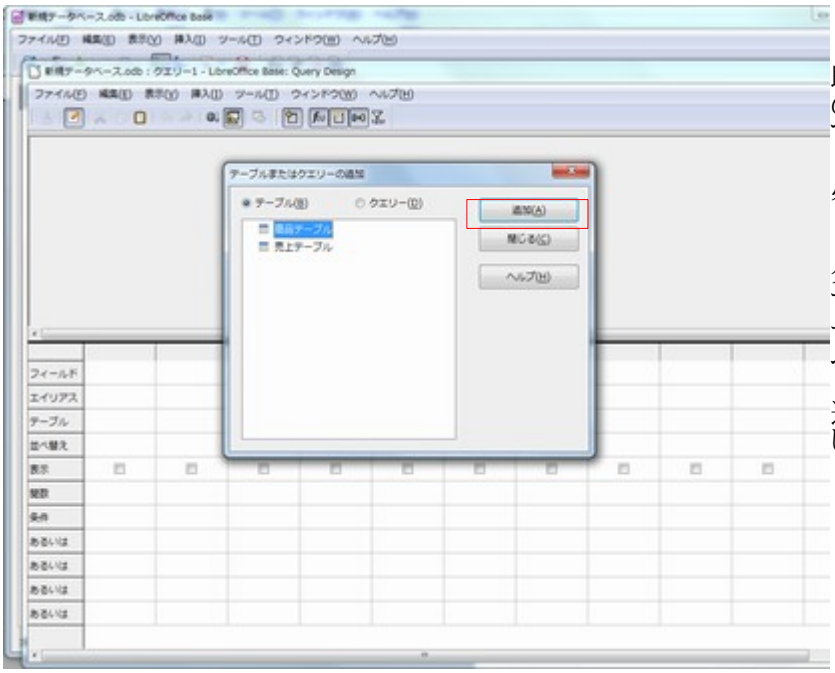
クエリー作成時には「どのテーブル同士を組み合わせれば欲しいデータが作成できるか?」を意識し、必要な項目を持つテーブルを決定し、テーブル同士がリレーションで結合されていることを確認します。
作成例では「売上テーブル」「商品テーブル」の2つのテーブルを使って売上明細表を作成します。

データベース画面からクエリーを選択します



右上のタスク欄にある
「デザイン表示でクエリーを作成」
をクリックします

「テーブルまたはクエリーの追加ダイアログ」が表示される画面に変わります



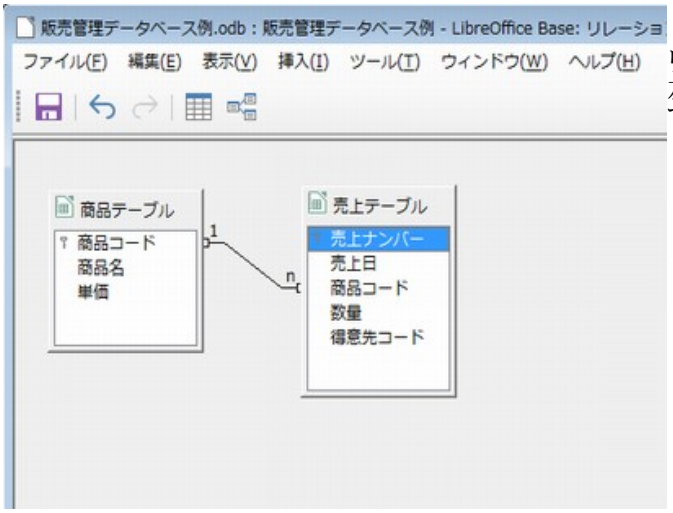
既定値でテーブルが選択されていますのでクエリーの元となるテーブルを選んで「追加」ボタンをクリックします

必要なテーブルの数だけ繰り返します

クエリーの元データにはクエリーを利用することも可能です。
その時はダイアログ内で「クエリー」を選ぶことでクエリーを選択できます

選択終了後、「閉じる」ボタンをクリックします

デザイン表示画面に選択したテーブルが表示されたことを確認します



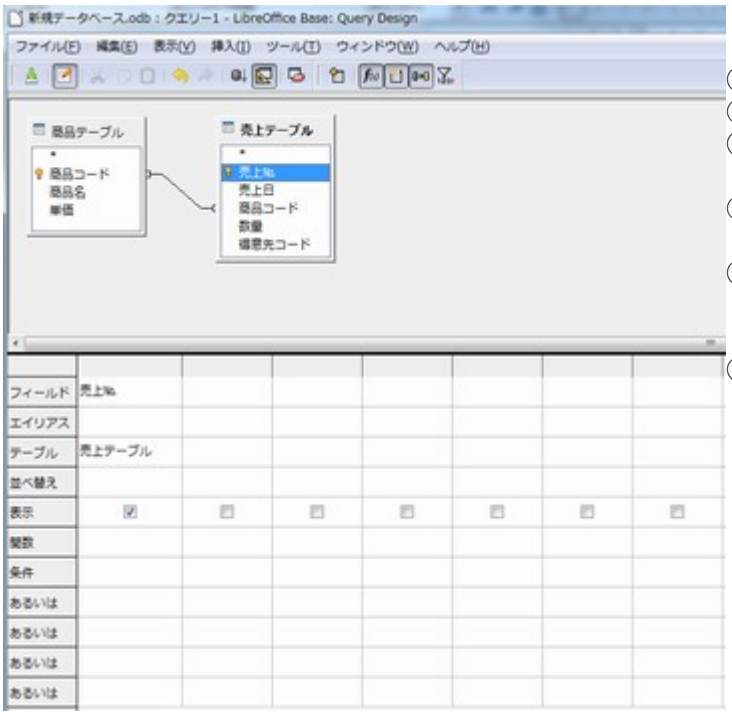
リレーションが設定されている場合
左のようにクエリーデザイン上でもリレーションを確認できます

取得したいデータは、こんな感じの表で出力してほしいので・・・・

売上No.	売上日	商品コード	商品名	数量	単価	売上額
1	6月1日	1222	えんぴつ	11	10	110
2	6月15日	1222	えんぴつ	10	10	100

1. 売上No.・売上日・商品コード・数量は「売上テーブル」から引っ張れば良いし・・・
2. 商品名と単価は「商品テーブル」から引っ張れば良いので・・・
3. 売上額は単価×数量で計算すればOKだな

というレイアウトを頭に浮かべて作業にかかります



①「売上No.」をマウスで選択しWクリック
②下段のフィールドに「売上No.」が登録されます
③同じように「売上日」・「商品コード」・「数量」をWクリックしていきます
④下段のフィールドにWクリックした項目が表示されていきます
⑤「商品名」・「単価」は商品テーブルにありますので商品テーブルの該当項目をWクリックします
⑥売上額は計算が必要ですから、フィールド欄に直接式を記入し、エイリアスに「売上額」と記入します

記入式: 単価*数量
乗算には*(アスタリスク)を使います
“ は自動記入されます

項目選択が終了したクエリー画面

フィールド	売上No.	売上日	商品コード	数量	商品名	単価	"単価" * "数量"
エイリアス							売上額
テーブル	売上テーブル	売上テーブル	売上テーブル	売上テーブル	商品テーブル	商品テーブル	
並べ替え							
表示	☑	☑	☑	☑	☑	☑	☑
関数							
条件							

フィールド：選択したフィールド名
エイリアス：別名　　上記では「単価×数量」を別名「売上額」で表示します
テーブル：フィールドが、どのテーブルに所属しているかを表示

ツールバーの「クエリーの実行（F5）」をクリックするとデーター覧が表示されます

クエリーの実行

ちゃんと別名で表示されています

	売上No.	売上日	商品コード	数量	商品名	単価	売上額
▶	1	6月1日	1222	11	えんぴつ	10	110
	2	6月15日	1222	10	えんぴつ	10	100

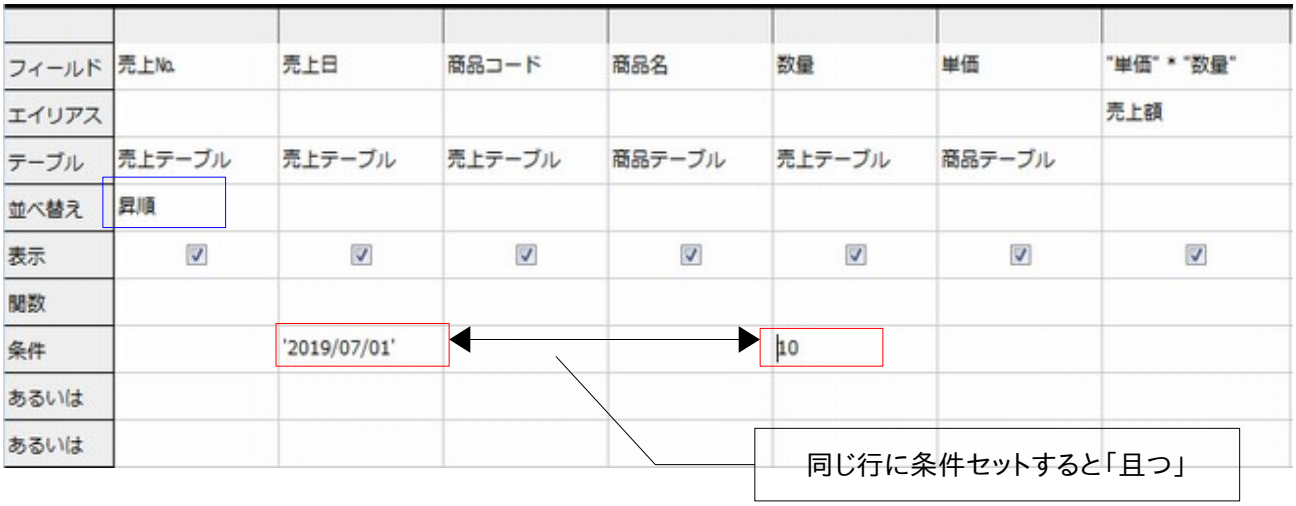
フィールドを移動させたい場合はフィールド名をクリックした後、移動先までドラッグします

フィールド	売上No.	売上日	商品コード	数量	商品名	単価	"単価" * "数量"
エイリアス							
テーブル	売上テーブル	売上テーブル	売上テーブル	売上テーブル	商品テーブル	商品テーブル	
並べ替え							
表示	☑	☑	☑	☑	☑	☑	☑
関数							

≪Point≫

算術演算子	意味
+	加算
-	減算
*	乗算
/	除算

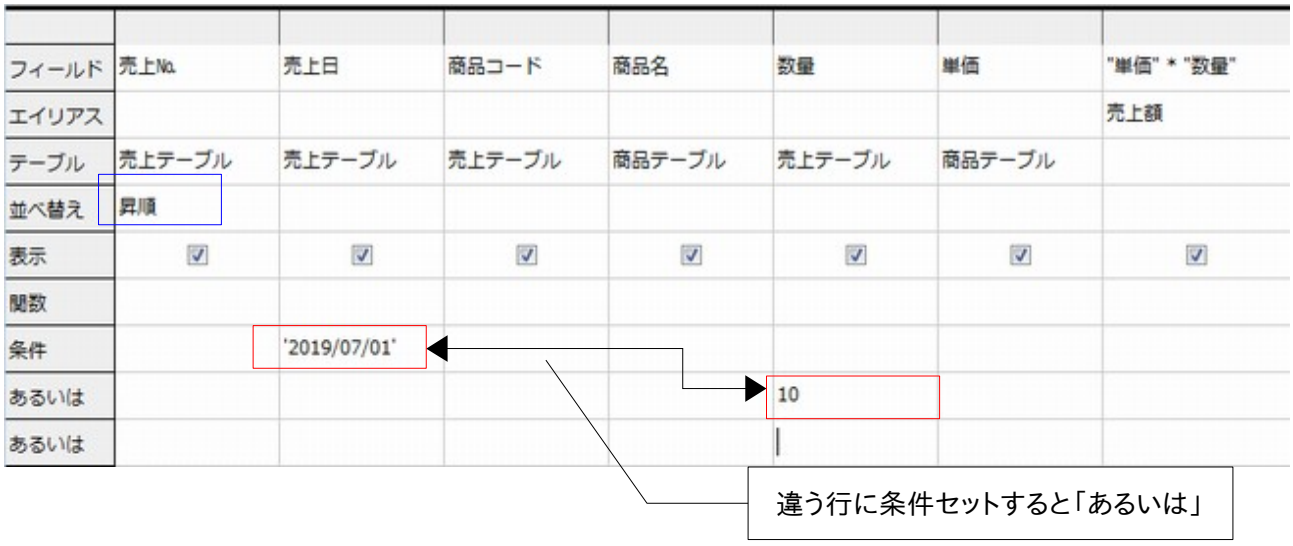
データ抽出する際は「並び替え」や「条件を付与」することができます



<上段の条件>
- 売上No.を昇順にする
- 売上日が 2019/07/01 のデータで、且つ単価が 10

	売上No.	売上日	商品コード	商品名	数量	単価	売上額
▶	1	2019/07/01	P00001	えんぴつ	10	10	100

※単価条件を 1 行下の「あるいは」欄に記載した場合



<上段の条件>
- 売上No.を昇順にする
- 売上日が 2019/07/01 のデータ、あるいは数量が 10 のデータ

	売上No.	売上日	商品コード	商品名	数量	単価	売上額
▶	1	2019/07/01	P00001	えんぴつ	10	10	100
	2	2019/07/01	P00001	えんぴつ	5	10	50
	6	2019/08/01	E00001	消しゴム	10	50	500

≪Point≫
テーブル:データの格納(データが保存されます)・・・データ保管庫として動作します
クエリー:データ加工(データは保存されません)・・・仮想表として動作します(クエリーとして保存は可能)

クエリーで利用する「フィルター演算子」

クエリーを利用する際によく使う「フィルタリング条件」には、さまざまな処理演算子や SQL コマンドが利用できます。
SQL コマンドには、関係演算子以外にも、クエリー処理を行うコマンドが各種存在しており、BASE のコマンド内でこれらを使用した場合は、BASE が自動的に対応する SQL コマンドに変換します。

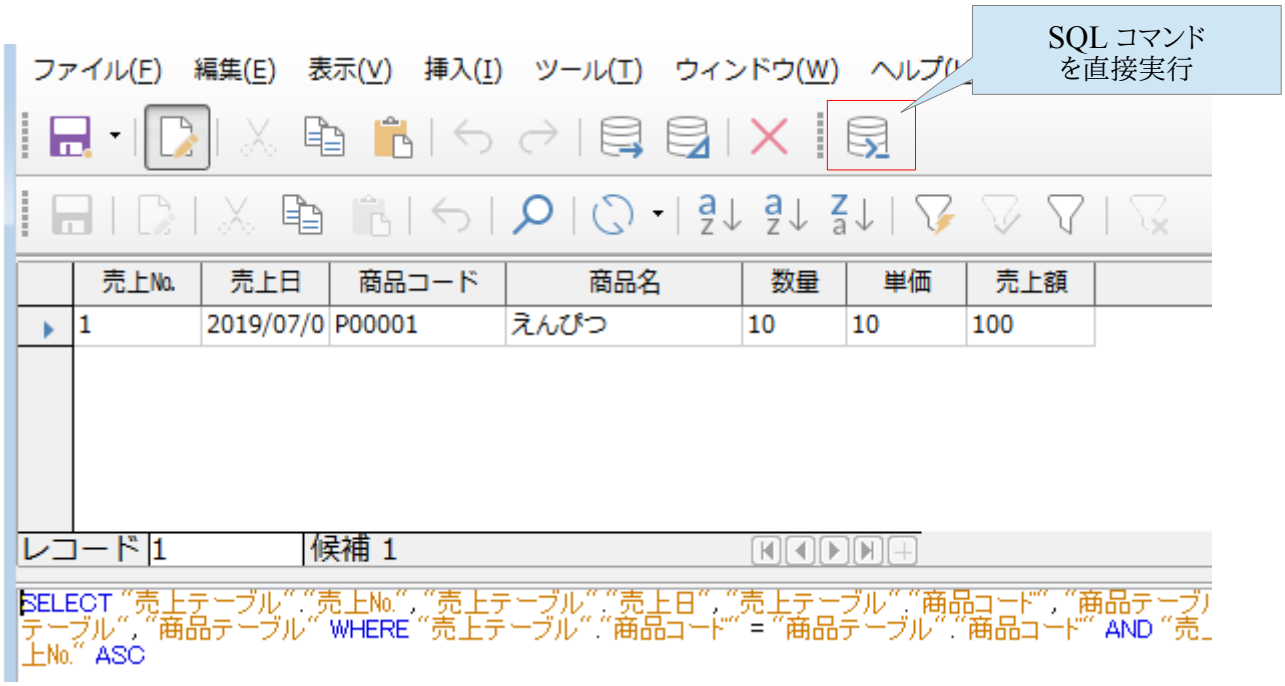
演算子	意味	条件が成立する場合
=	等しい	指定値とフィールド値が一致する。
<>	等しくない	指定値とフィールド値が一致しない。
>	大きい	指定値よりもフィールド値が大きい。
<	小さい	指定値よりもフィールド値が小さい。
>=	大きいか同等	指定値とフィールド値が同等か、その方が大きい。
<=	小さいか同等	指定値とフィールド値が同等か、その方が小さい。

BASE コマンド	SQL コマンド	意味	条件が成立する場合
IS EMPTY	IS NULL	空(NULL)	データフィールドが空です。
IS NOT EMPTY	IS NOT NULL	空でない	データフィールドが空ではない。
LIKE（任意の文字数のワイルドカード文字は* 1 文字のワイルドカード文字は ? ）	LIKE (任意の数の文字に対応する% プレースホルダー) 一文字のワイルドカード文字 _ ）	検索パターンと一致する	指定値がフィールド値に含まれている。ワイルドカード文字「*」を利用した「あいまい」検索を実施します。ワイルドカード文字「*」および「%」は複数の任意の文字です。LibreOffice の疑問符「?」や SQL クエリーの下線「_」は、任意の一文字を表します。
NOT LIKE	NOT LIKE	検索パターンと一致しない	指定値がフィールド値に含まれていない。
BETWEEN x AND y	BETWEEN x AND y	区間 [x,y] に存在する	データフィールドの値は x 値と y 値の間の値である。
NOT BETWEEN x AND y	NOT BETWEEN x AND y	区間[x,y]には存在しない	データフィールドの値は x 値と y 値の間の値ではない。
IN (a; b; c...) 全ての値リストで区切りに使っているセミコロンに注意してください。	IN (a, b, c...)	a, b, c..を含む	指定値 a, b, c,...のどれかがフィールド値に含まれている。多くの任意の値を対象にすることができ、クエリーの結果は論理「OR」で求められます。a, b, c... といった値は数値でも文字でもかまいません
NOT IN (a; b; c...)	NOT IN (a, b, c...)	a, b, c..を含まない	指定値 a, b, c,...のどれかがフィールド値に含まれていない。
= TRUE	= TRUE	「True」値	「True」値がフィールド値に含まれている。
= FALSE	= FALSE	「False」値	「False」値がフィールド値に含まれている。

《使用例》

```
SELECT * FROM "売上データ" WHERE "処理日" IS NULL
SELECT * FROM "売上データ" WHERE "処理日" IS NOT NULL
SELECT "商品名" FROM "商品マスタ" WHERE "商品名" LIKE '%PC%'
SELECT * FROM "売上データ" WHERE "処理日" BETWEEN '2019/04/01' AND '2019/06/10'
SELECT "商品 ID", "商品名", "グループ名" FROM "商品マスタ" WHERE "商品 ID" IN ( 5, 7, 9 )
```

Access に無い処理「SQL コマンドを直接実行」とは？



「SQL コマンドを直接実行ボタン」がオンの時は接続しているデータベースシステムで直接実行されます
このボタンがオフの時は一旦 LibreOffice で確認後、実行されます

連結する「Union クエリー」や「Case 演算子を利用した条件分岐」は「SQL コマンドを直接実行ボタン」がオンの状態でなければエラーになります。
このことは LibreOffice で実行できるクエリー（SQL コマンド）には制限があることを示しています。
SQL 構文に間違いが無いのに「エラー」になる場合は、LibreOffice の制限に引っかかっている可能性がありますので、「SQL コマンドを直接実行ボタン」をオンの状態にして再実行してみてください。

【豆知識】
BASE は組込み DataBase によって SQL 構文を使い分ける必要があります。
Firebird を組込 DB にしている場合は CASE 文を利用できますが、Access-mdbファイルを利用した場合、CASE 文はエラーになります(.accdb ファイルの場合は不明です)
mdb ファイルを利用した場合は IIF 文か SWITCH 文を利用します。

≪CASE 文の例≫
SELECT "担当者名", CASE WHEN "性別" = 0 THEN '女性' WHEN "性別" = 1 THEN '男性' ELSE NULL END "性別" FROM "担当者マスタ" ORDER BY "担当者名"

≪IIF 文の例≫
IIF("所属" ='A','優良','良です') AS "判定"

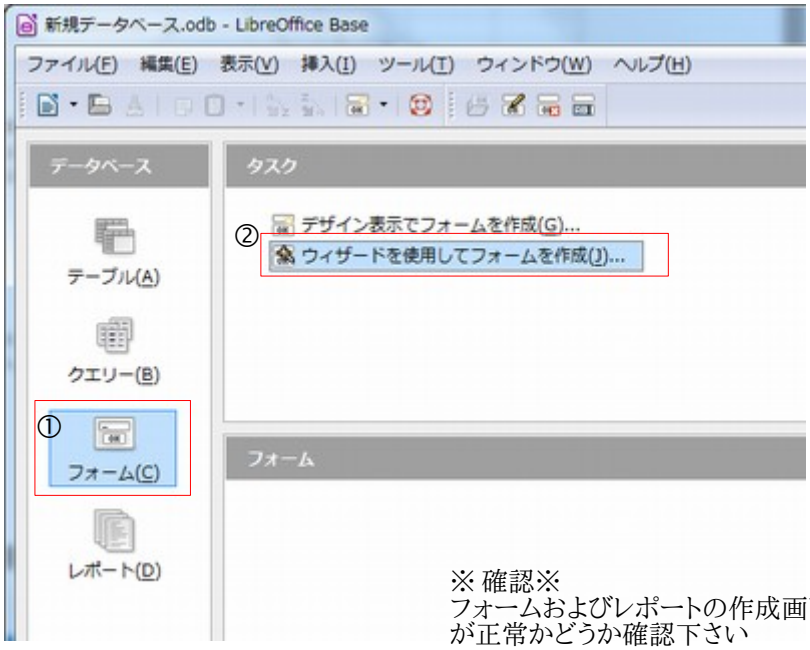
≪SWITCH 文の例≫
SWITCH("所属" ='A','優良',TRUE,'良です') AS "判定"

フォームの作成

フォームとはデータを効率よく入力したり、更新したり、条件を設定したりする為のオブジェクトで、必ず作成しなければならないオブジェクトではありません。

データ入力はテーブルやクエリーでも可能ですが、入力項目が多い場合には 1 レコードが横に長くなりすぎてしまいデータ位置を間違えるなど人為的ミスが起こりやすくなります。
フォームを作成すると 1 レコードを全て 1 画面に表示することが可能となり、入力ミスの軽減や入力の効率化が可能となります。

≪ウィザードを利用してフォームを作成してみる≫



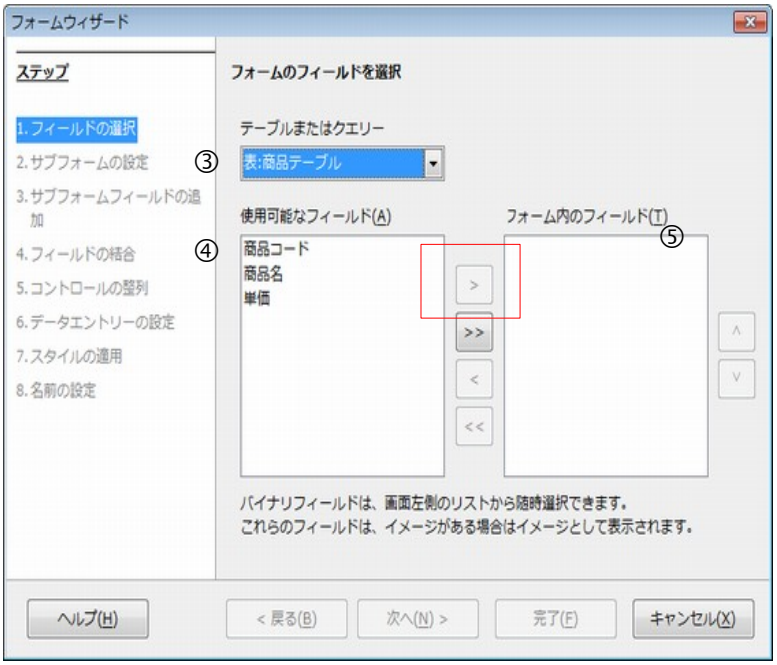
①フォーム(C)を選択

②ウィザードを使用してフォームを作成(J)を選択

③フォームの元になるテーブルやクエリーを選択します
④どのフィールドを配置するのか?を選択します

⑤フォーム内のフィールド(T)に送りこみます

※ 確認※
フォームおよびレポートの作成画面に遷移しない場合、Java のインストールが正常かどうか確認下さい



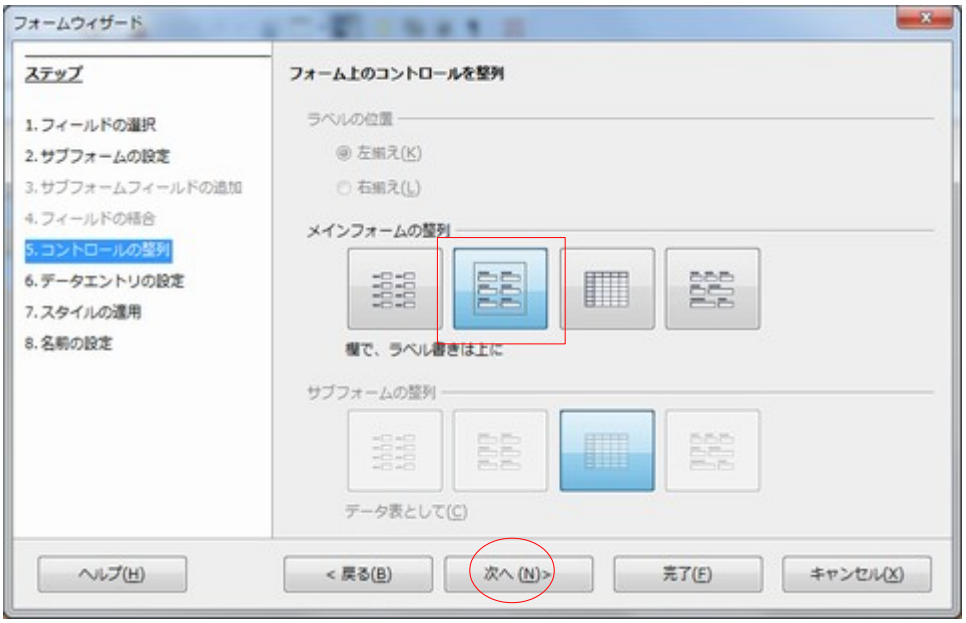
「次へ」
ボタンをクリックします

※Access で対応している「タブフォーム」の作成はできません

サブフォームを設定する画面になります

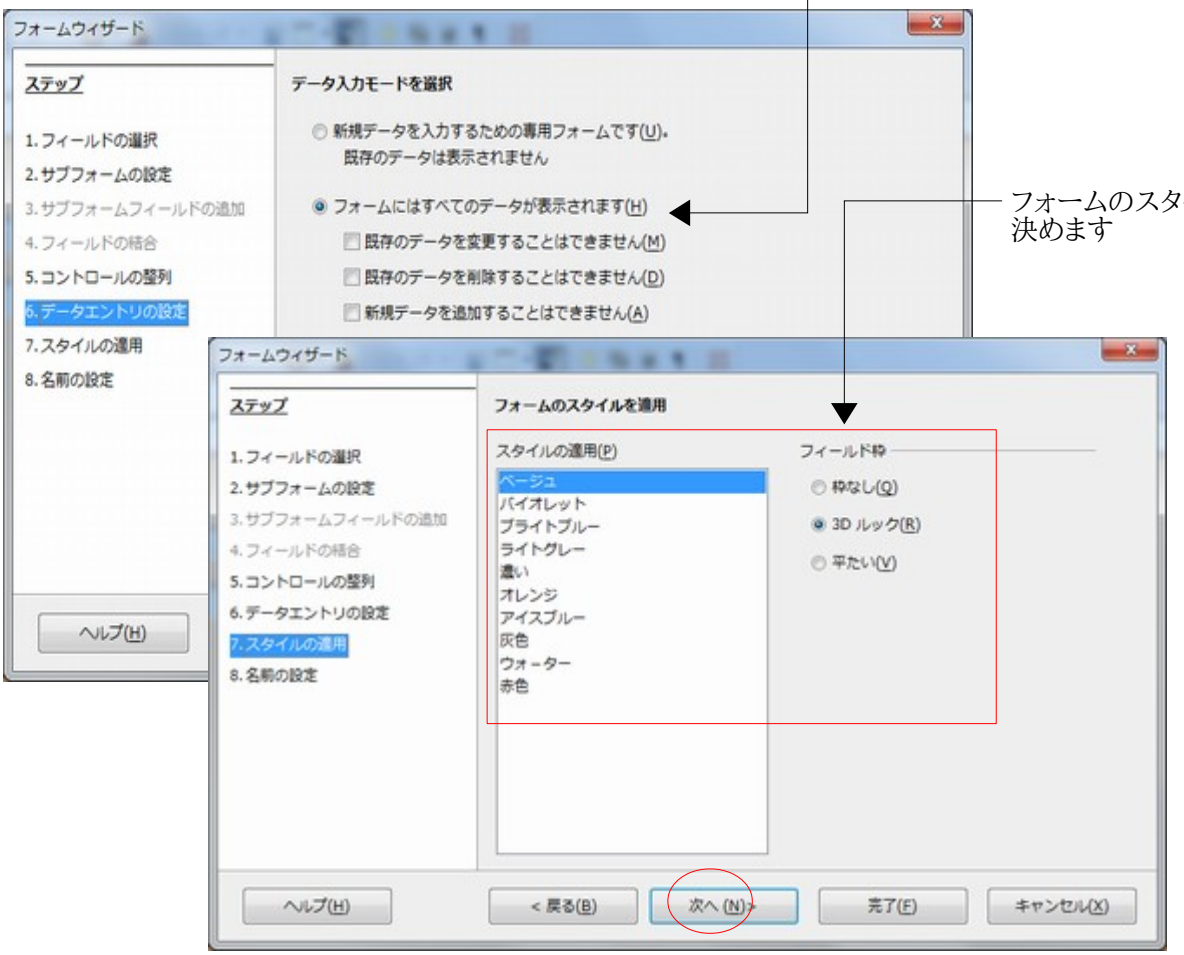
サブフォームは設定しませんので「次へ」とクリックします

フォーム上に配置するコントロール位置を指定します



好きな整列を選んでかまいません。選択を変えるとダイアログ背後のフォームも連動して変わることを確認してください。

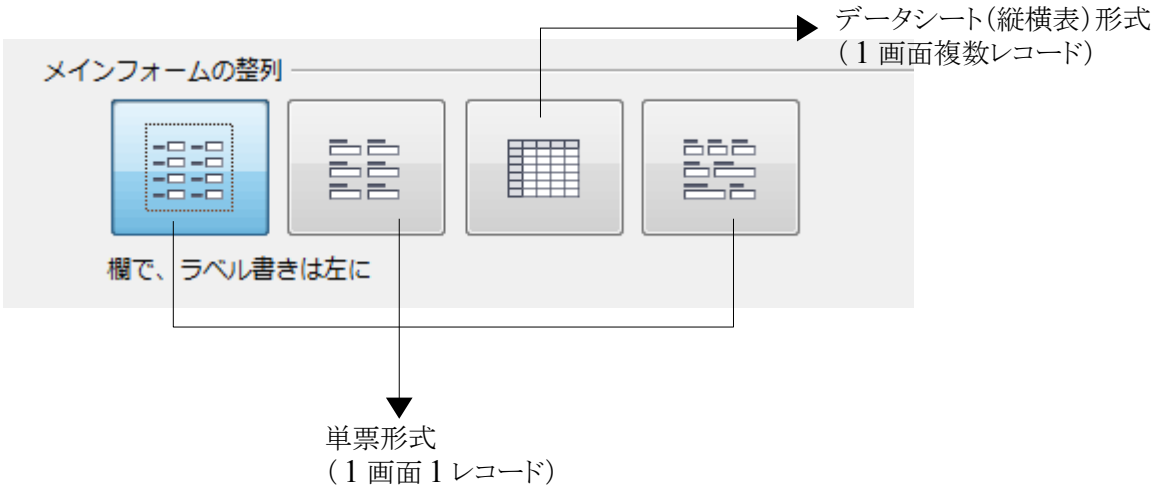
通常：「フォームには全てのデータが表示されます」を選択します



フォームのスタイルを決めます

最後にフォームの名前を決めて完了です

≪フォームのレイアウトについて≫

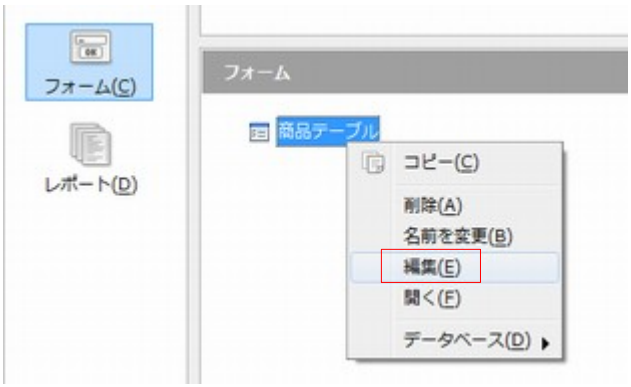


フォーム適用の目安
≪単票形式が良いもの≫
入力項目が多く、データシートでは入力する際に横スクロールが発生するもの・・・・・顧客マスターなど

≪単票形式または単票形式+サブフォームが良いもの≫
売上と売上明細のように従属した別データを同時に登録する必要があるもの・・・・・販売管理画面など

≪データシートが良いもの≫
入力項目が少なく、データ全体を一覧で把握したほうが効率が良いもの・・・・・店舗マスターなど

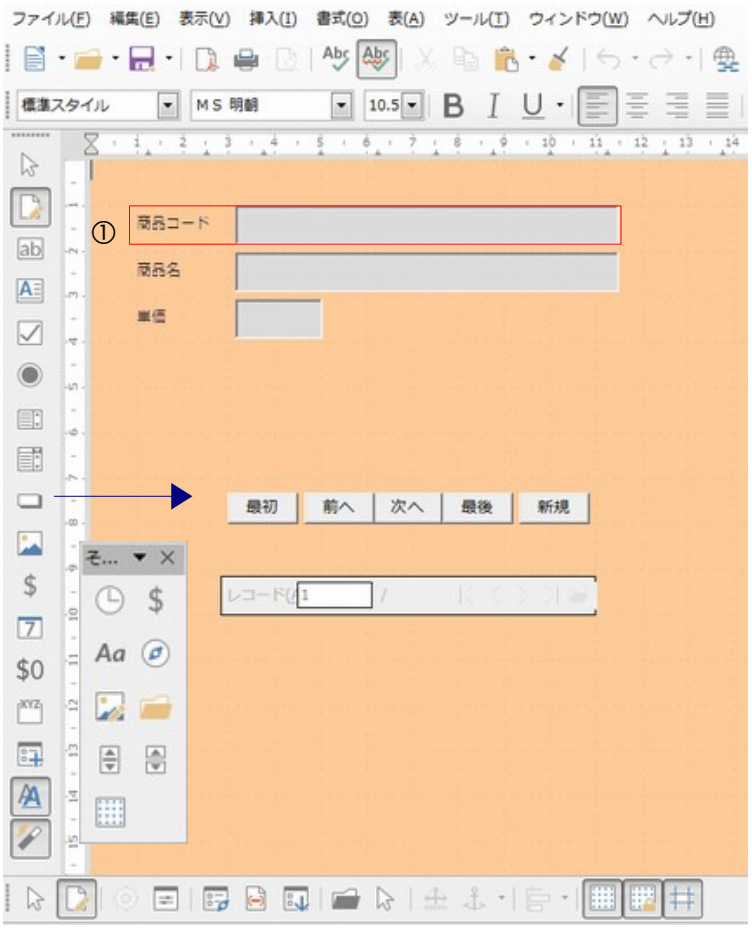
≪フォームの編集方法≫



フォーム名を選んで
右クリック→編集

編集で良く使うフォームデザインツールバー　「フォームボタンとフォームナビゲーターボタン」

フォーム画面（運用画面：通常モード）

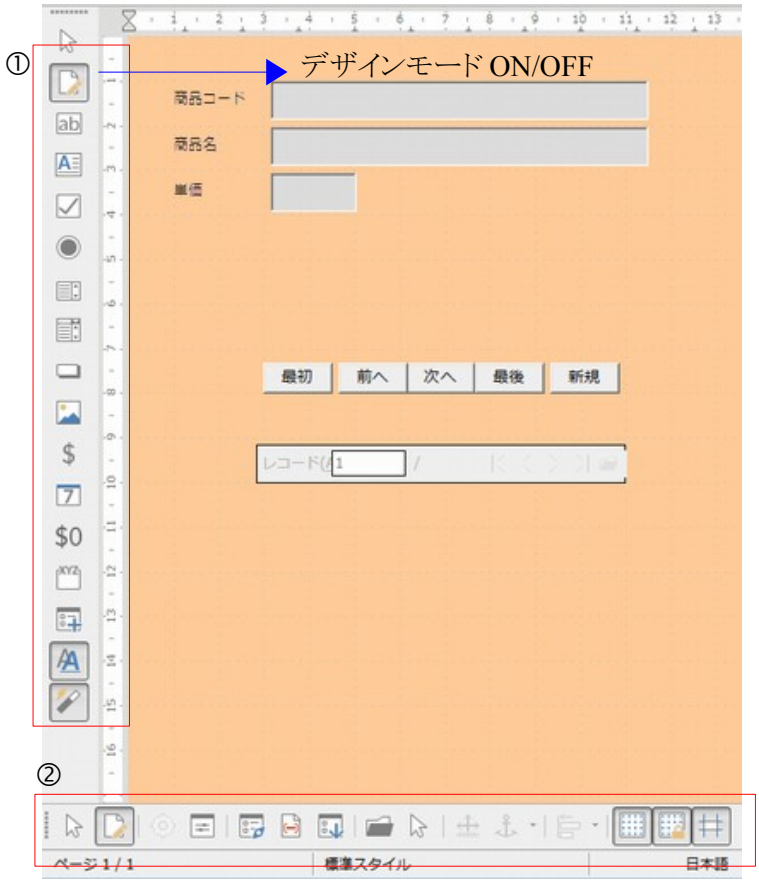


①テキストボックス（文字や数値などのデータを入力するオブジェクトです）

②以前は標準でレコード移動ボタン（前後のレコードを表示する際に利用します）が表示されていましたが、表示されなくなりました。
対策1
・ボタンを配置して「アクション」指定
対策2
その他のコントロールを表示し
・ナビゲーションバーを貼り付け

表示→ツールバー
フォームナビゲーションでも可

フォーム画面（編集画面：デザインモード）

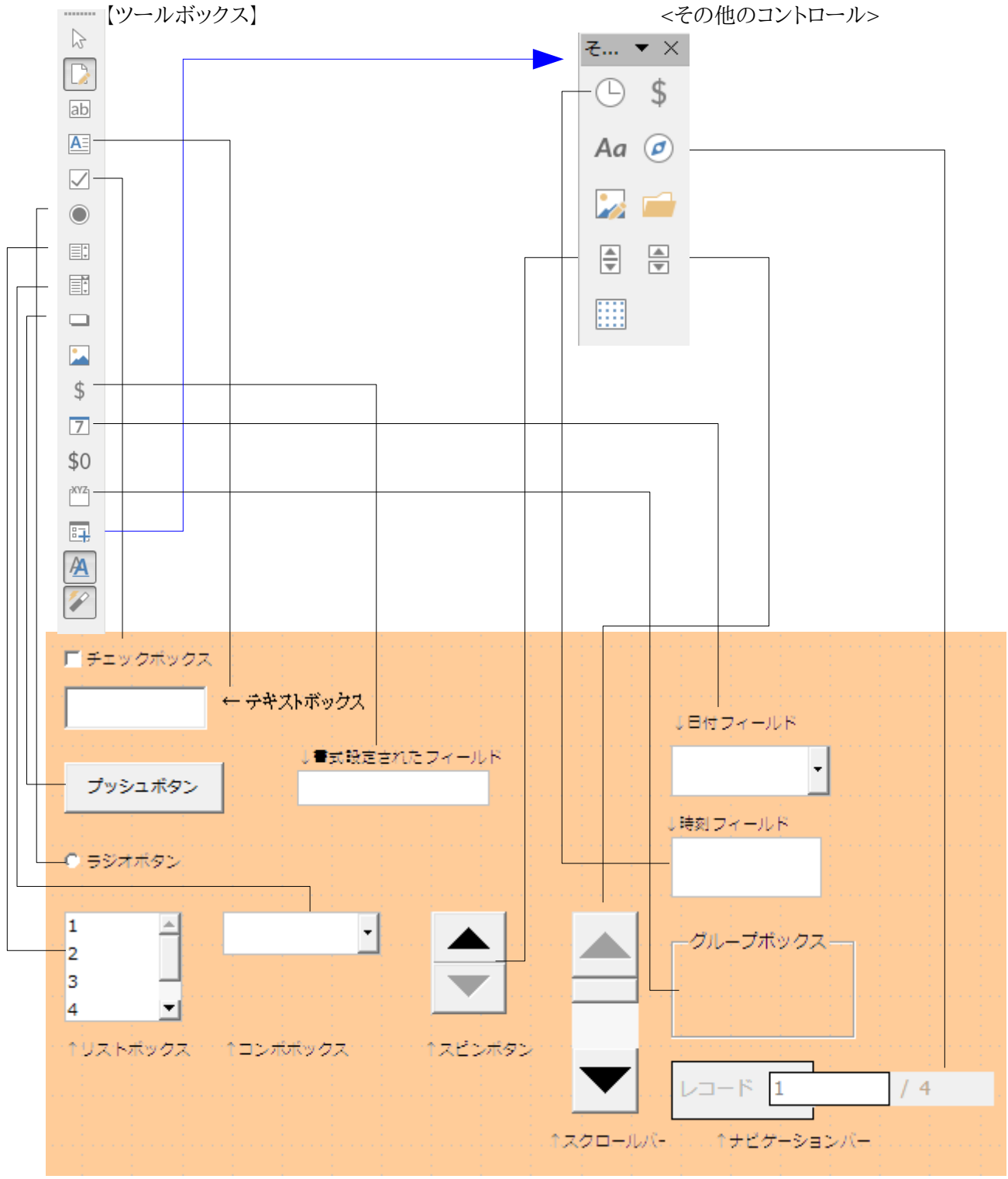


①ツールボックス（各種コントロールがセットされています）

②プロパティ（属性）を変更するツールバー
データソース変更
各種イベント
タブコントロール　etc

フォームのコントロール

フォームでは、データの入力や参照を行う際に、コントロールと呼ばれるさまざまなオブジェクトをフォーム上に貼り付けて利用します。レコードの移動などもコントロールを利用して操作した方がスムーズに行える場合がほとんどです。ココではいろいろなコントロールをチェックしてみましょう。



コントロール説明

コントロール名	主な用途
チェックボックス	チェック項目の入力「はい・いいえ」に利用します
テキストボックス	氏名・住所などテキストを入力する場合に利用します
ラジオボタン	幾つかの中から１つを選ぶ場合に利用することが多いです
リストボックス	あらかじめ表示されたリストの中からいくつか選択する場合に利用します
コンボボックス	リストへの入力が可能なリストボックス（1つ選択可能）と考えてもらえば良いです
書式設定されたフィールド	Calc のセルのように書式を設定できるようです（３桁区切り¥マーク付数字など）
スピンボタン	値の増減などをマウスクリックで行なう場合に利用します
スクロールバー	スピンボタンよりも大きな値変化をマウス操作で行う場合に利用します
グループボックス	ラジオボタンなどを並べて１つのグループとして利用する際に使います
ナビゲーションバー	レコードの移動を操作するコントロールです
日付・時刻フィールド	日付や時刻を専用に受け付けるテキストボックスです（非連結でもカレンダーコントロールが利用できます）

【Base と Access との機能比較】

1. Base では、テキストボックス入力時に日本語 IME を ON・OFF することはできません。（Access は可能です）・・・OS が Windows の場合マクロで ON・OFF 実装可能
2. Base ではプライマリーキーを設定しない場合、データを追加・削除できません。
（SQL の CreateTable 文を発行後、手動設定することもできませんので注意してください）
3. 起動時に特定のフォームを起動させる場合は別途マクロを作成し、イベント登録が必要です。
4. ウィザードで作成できるサブフォームは１つまでです。（手で作成すると２つ以上も可能）
5. データのインポート及びエクスポートコマンドがありません。・・・マクロで実装可能
6. クエリーの条件としてパラメーター値の設定は可能です。（パラメータークエリー）
7. フォームの値を直接クエリーのパラメーターとして SQL 文に反映させることはできません。（マクロによる SQL 再作成が必要です）
8. フォームにヘッダーおよびフッターを設定することができません。
9. クエリーを実行する際にマクロで作成したユーザー定義関数（Function 関数）を利用することはできません（Java/C で作成したユーザー定義関数（UDF）なら実行可能らしい？）
10. テーブルに格納されたデータでも ODB ファイルを保存するまではデータが保存されていない。

【確認されている不具合など】　～Firebird になって安定度が増したとは言うものの～

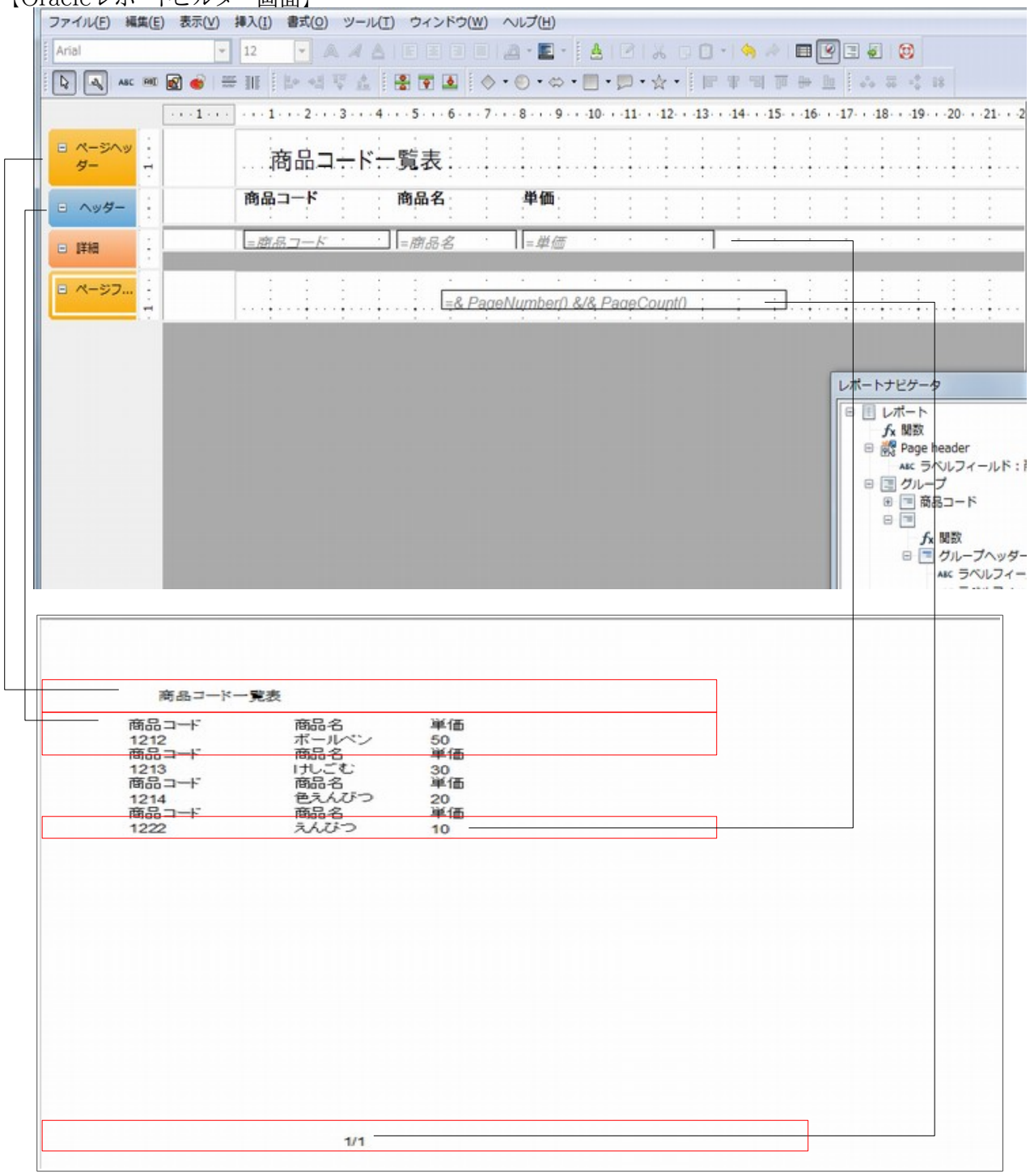
1. データを Calc にコピー貼り付けすると日本語が文字化けします。（ドラッグ&ドロップなら大丈夫です）
2. Calc や Excel ファイルなどの表計算ファイルをリンクテーブルとする場合は１データベースに１テーブルの利用となり、複数リンクテーブルを設置することができません。
3. レポート作成用レポートビルダーにエラーが多く、ヘッダーやフッターでの表示不具合が発生する（場合によってはレポートが開かずにファイル終了になる場合がある）・・・Ver6.0 以降は安定しているが今後バージョンアップ後に不安定になる可能性がある。
4. 保存したのにテーブルデータが保存されていない場合がある。・・・Ver6.0 以降は安定している

パラメータークエリーの例
SELECT * FROM "人物テーブル" WHERE "よみがな" = :「よみがな」を入力

レポートを作成してみる

レポートとはデータを印刷する為のオブジェクトで、出力したい帳票レイアウト(タックシールは Writer の差込印刷で作成します)を作成し、並び替えや非表示設定などもおこなうことができます。(グループ化や小計・合計表示も可能です)
Base でレポートを作成する際に利用するツールはオラクルレポートビルダー(Oracle-Report-Builder)です。基本的に Access のレポート作成と同じ感覚で作業できますが、Access のレポート作成機能と比較すると機能不足だけでなく不具合も多く Access でできる事が全てできるわけではありません。

【Oracle レポートビルダー画面】



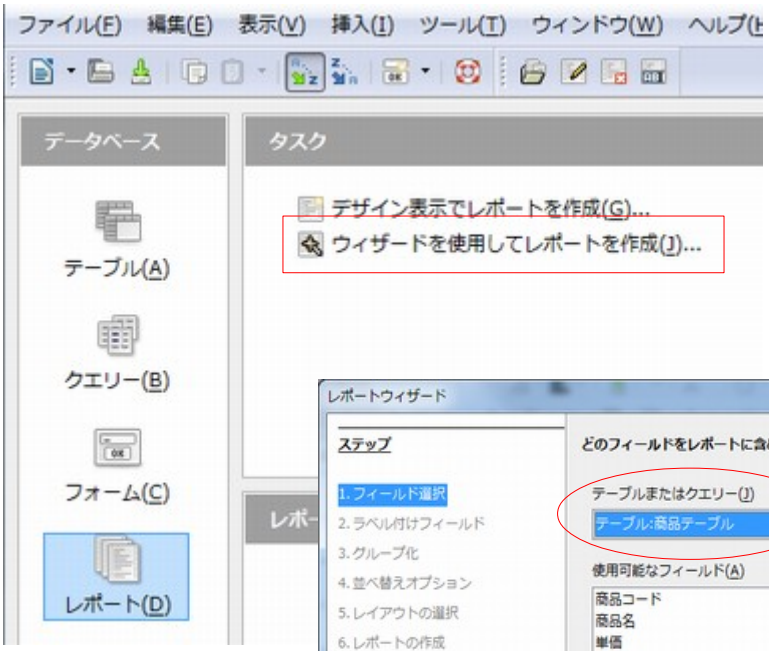
用語解説

1. ページヘッダー:レポート印刷時に各ページの先頭に印刷されるセクションです(サブタイトルや小見出しに利用します)
2. 詳細:各レコード(データ)が表示されるセクションです
3. ページフッター:レポート印刷時に各ページの下部に印刷されるセクションです(日付やページ番号などを印刷するときに利用します)
4. レポートヘッダー:レポートを印刷した時に最初のページの先頭に印刷されます(既定では未設定)
5. レポートフッター:レポートを印刷した時に最後のページの末尾に印刷されます(既定では未設定)
6. グループヘッダー&フッター:グルーピングを行った場合に設定されます(個人毎に帳票を出力する場合などには必須設定事項です)

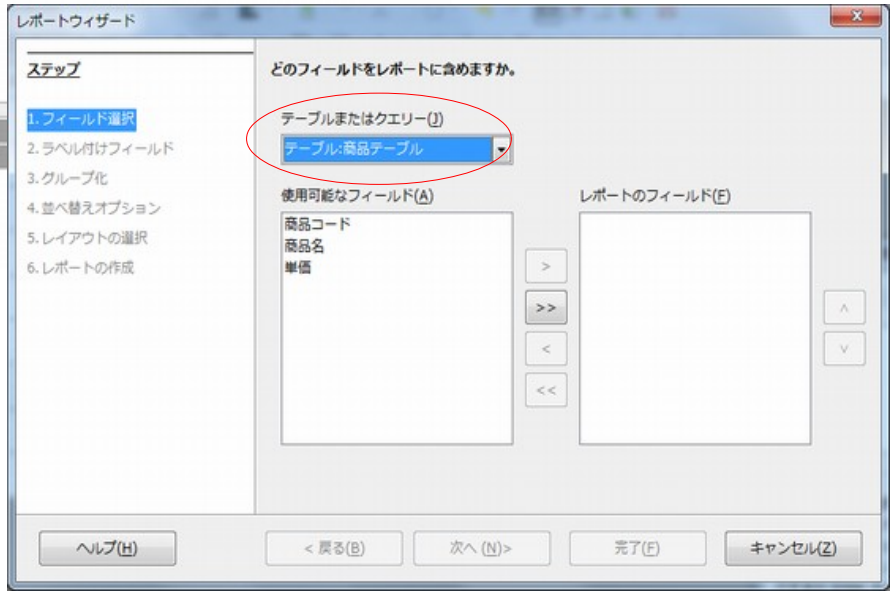
Access から移行する際に問題とした事項(レポート)

1. レポート行の高さを低め(狭く)設定する際の調整幅が少ない。
2. テキストボックスの枠を印字することができない。
3. ページフッターに合計件数や合計金額を表示することができない。
4. 詳細に表示するデータが多いとき、グループフッター部分に詳細データが食い込んでくる場合がある。
5. レポート表示に時間がかかる。(Javaだから?)・・・・Ver6.0 以降かなり改善されたが・・・
6. レポートで利用できる関数が少ない。(Sum や Count が使えない:集計は設定で実施できる)
7. フォント変更がうまく反映されない場合がある。

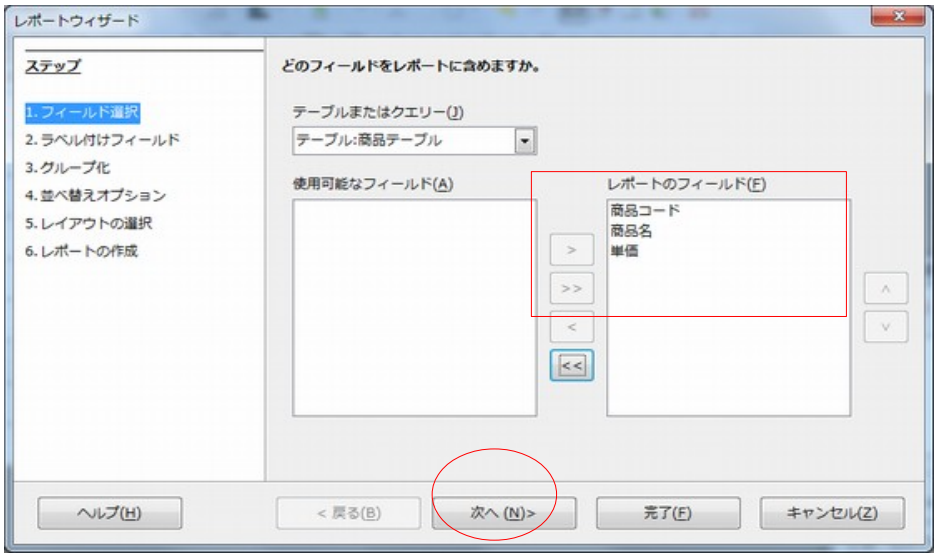
【ウィザードを使ってレポートを作成する】



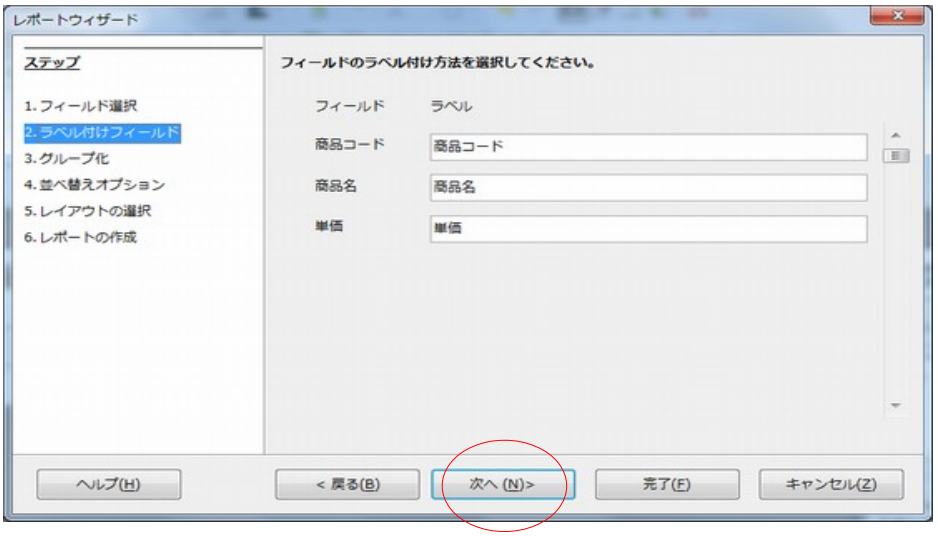
1. ウィザードを使用してレポートを作成(J)を選択します
2. テーブルまたはクエリーを選びます



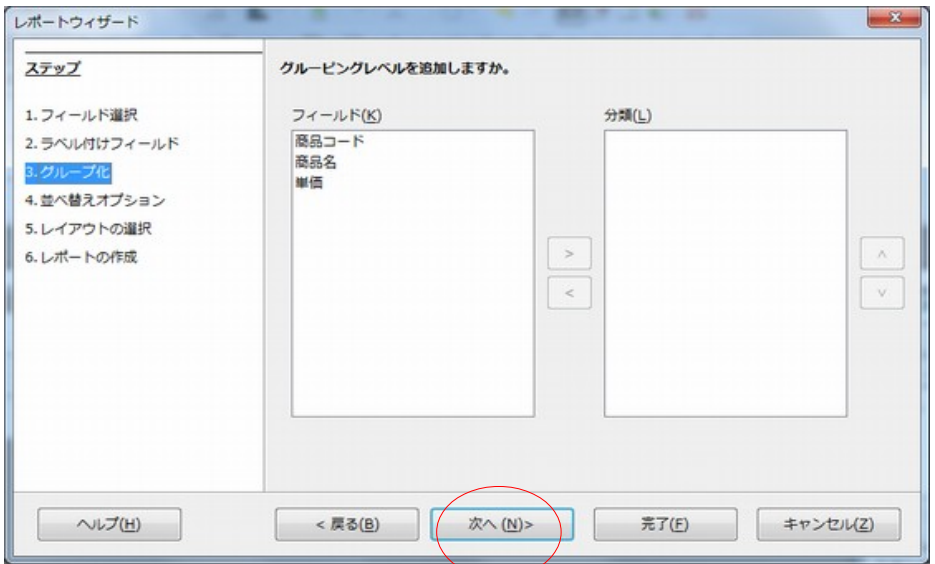
3. レポートに表示したいフィールドを選択します



4. どのような項目名で表示するのか?を問いかけています(変更しない場合は次へを選択します)

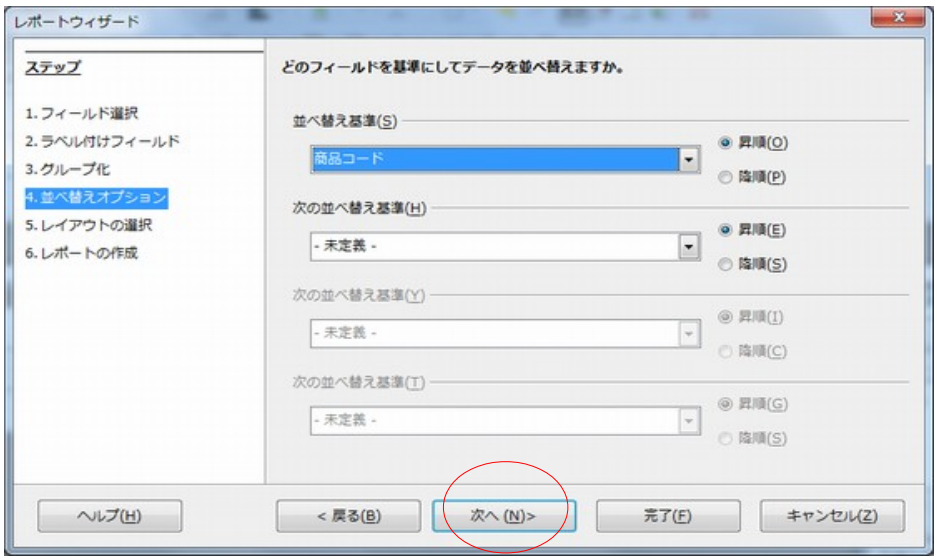


5. グルーピング(グループ化)設定(特にない場合は次へを選択)

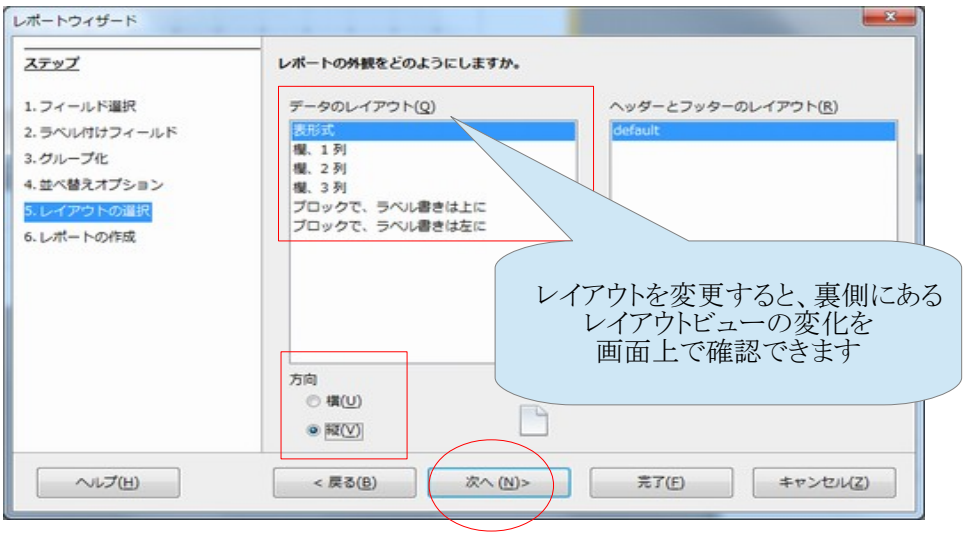


※P34 【レポートにおけるグルーピング(グループ化)の利点】参照

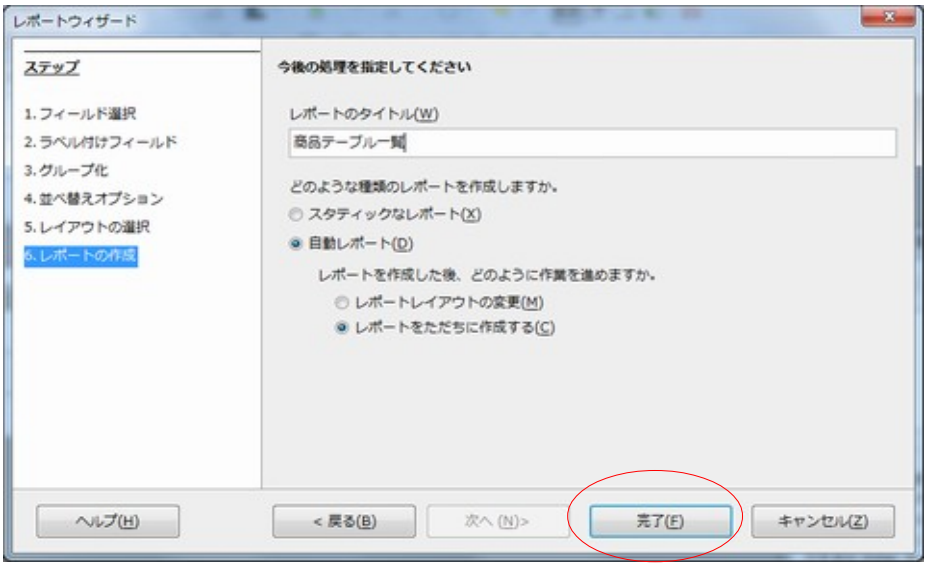
6. 並べ替えの設定（例では商品コードを昇順に並び替えています）



7. レポートのレイアウトを指定します（例では「表形式」で縦方向のレポートを指定しました）

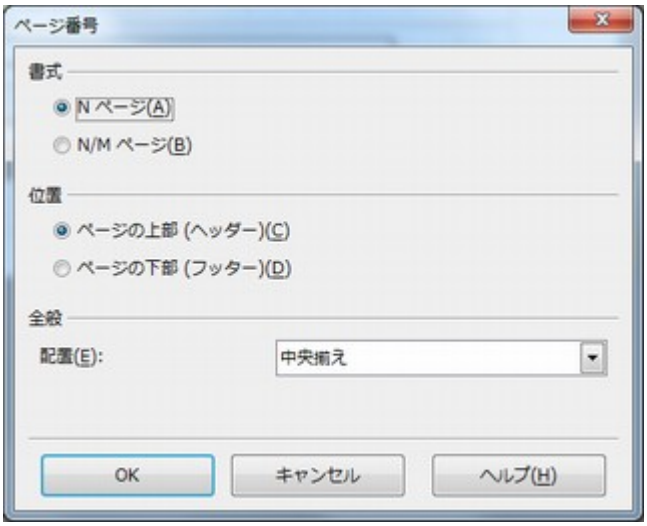


8. レポート名を指定して、レポートをただちに作成する（C)を選択し「完了」でレポートが作成されます



【ページ番号や日付・時刻の挿入方法】
レポートを右クリック→編集を選択して、レポートビルダーを開きます

メニュー→挿入→ページ数



≪書式≫
- N ページ：　1
- N/M ページ：　1/4

≪位置≫
ページの上に表示（ヘッダー）
ページの下に表示（フッター）

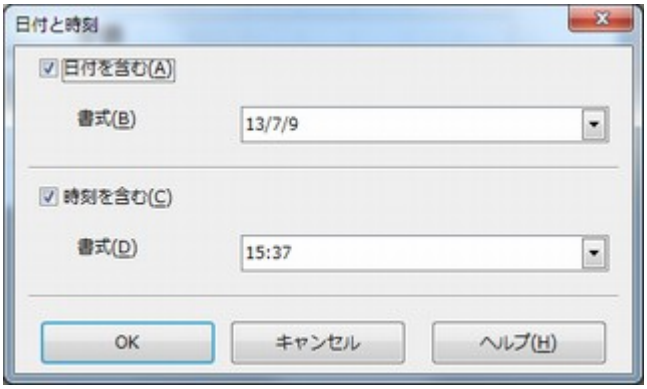
エラーポイント!（Ver4.0 から未修正のバグ）
挿入された状態ではページ表示するとエラーになります、手作業で修正してください（「ページ」を手で削除します）　～エラーにはならないが表示がおかしい～

=& PageNumber() ページ &/& PageCount()

=& PageNumber() &/& PageCount()

修正は　右クリック　→　プロパティ　を表示させ、データタブのデータフィールド　にて行います

【日付と時刻の表示】
メニュー→挿入→日付と時刻



【レポートにおけるグルーピング（グループ化）の利点】　～重要～
グルーピング（グループ化）は同じ種類や分類（店舗や期間等）で一括りにする集計単位と考えることができます。レポートでグルーピング（グループ化）を行うと、各グループごとにヘッダー・フッターを置くことができ、グループ毎に見出しをつけたりグループ毎に集計を行うことができるため、単にデータレコードを羅列しただけのレポートではなく、データを一定の視点から整理した結果を表現することが出来ます。