BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan, Koordinasi, dan Divisi

3.1.1 Divisi

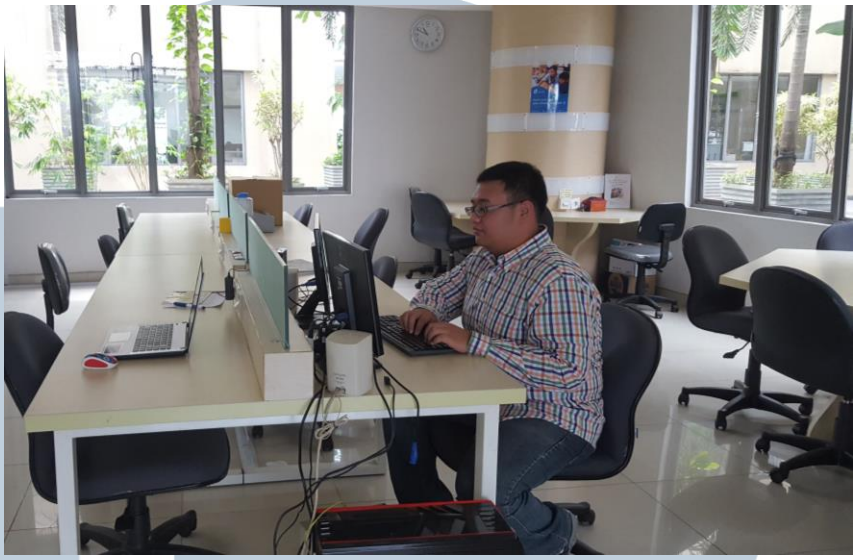
Pada kesempatan magang di *VBS Technology* ini penulis ditempatkan pada bagian *Technical Department* dimana karena perusahaan ini juga masih sebuah perusahaan *start up* sehingga *owner* sekaligus *supervisor* penulis merupakan orang yang sama.

Gambar 3.1 Penulis di depan ruang kerja

3.1.2 Kedudukan dan Koordinasi

Selama melakukan masa magang penulis mendapat kedudukan sebagai *Technical Staff* yang bertugas melakukan *debug* pada sebuah *software* yang sudah ada.

Dalam melakukan tugas - tugas yang penulis dapat, penulis sering berkoodinasi langsung dengan *superivsor* baik untuk bertanya mengenai pemecahan masalah ataupun berdiskusi memikirkan fitur yang akan ditambahkan.



Gambar 3.2 Penulis sedang bekerja

3.2 Tugas-Tugas yang Dikerjakan

Selama melakukan kegiatan magang pada *VBS Technology* selama 2 bulan ini, penulis mendapat pekerjaan utama yang merupakan mencari *bug* pada *program Point Of Sales,* selain itu penulis juga mendapatkan beberapa pekerjaan selain pekerjaan utama penulis yang penulis, pekerjaan tambahan yang penulis dapat dan kerjakan antara lain adalah

1. Membuat *table* baru pada *database* yang sudah ada yang akan digunakan untuk menambahkan fitur pada *program* yang penulis kerjakan
2. Membuat *trigger* pada suatu table yang akan aktif ketika ada data baru yang masuk ke table yang memiliki *trigger* tersebut
3. Membuat *Stored Procedur* pada *database* yang digunakan untuk meringankan beban program ketika digunakan untuk melakukan *query*
4. Memasukkan data pada *program* yang data tersebut merupakan *data* asli yang akan digunakan oleh *client* yang akan menggunakan *program* ini
5. Menambahkan menu, *form*, laporan baru pada *program*
6. Mengubah beberapa *modul* yang digunakan pada *program*

7. Mengubah *class* yang digunakan untuk melakukan koneksi ke dalam *database*

Dalam menjalankan tugas - tugas yang diberikan tersebut, penulis menghabiskan minggu awal proses magang untuk memahami *coding* yang sudah ada para program *Point Of Sales* yang penulis kerjakan, karena sebelumnya sudah ada yang membuatnya, jadi untuk mencari dan memperbaiki *bug* dari program tersebut tentu memahaminya terlebih dahulu, ditambah dengan cara koneksi yang berbeda serta penggunaan dari *stored procedure* yang baru pertamakali penulis lihat dan akan penulis gunakan dalam pengerjaan tugas yang penulis dapat selama magang ini.

Pada minggu - minggu seterusnya penulis sudah mulai mengerti dan dapat menyelesaikan satu persatu tugas yang diberikan kepada penulis, dan pembenaraan dari *bug* yang penulis kerjakan akan dicoba terlebih dahulu oleh *supervisor* sekaligus pemiliki dari *VBS Technology* dan teman magang penulis yang ditugaskan juga untuk melakukan *testing* untuk membantu dalam menemukan *bug* juga.

3.3 Hasil Kerja

Proses melakukan kerja magang dimulai pada Rabu tanggal 13 Juli 2016, pada minggu awal proses kerja magang, penulis lebih sering mendapatkan *briefing* dari *supervisor* penulis mengenai perusahaan, dan *project* yang akan penulis kerjakan selama melakukan kegiatan magang pada *VBS Technology*.

Seperti yang sudah penulis sebutkan sebelumnya, pekerjaan utama yang penulis kerjakan adalah mencari dan memperbaiki *bug* pada *program Point Of Sales,* selama melakukan kegiatan magang cukup banyak *bug* yang penulis temukan pada *program Point Of Sales* ini, dan dari semua *bug* yang telah penulis temukan ini tidak semua dapat langsung diperbaiki banyak juga yang dalam menyelesaikannya penulis membutuhkan bantuan dari *supervisor* penulis, berikut merupakan sebagian *bug* yang penulis temukan selama melakukan kegiatan magang pada *VBS Technology* ini baik *bug*

yang memang sudah ada atupun *bug* yang ada karena kesalahan penulis dalam membuat suatu fungsi atau *form*, antara lain adalah

(a) Kesalahan dalam menampilkan harga produk pada *form* transaksi pembelian.

(b) Setelah mencari produk pada *sale*, *data* yang terpilih tidak semuanya masuk ke dalam tempat yang dituju.

(c) Saat menyimpan transaksi terdapat kesalah nilai yang terambil pada *form sale*, jadi *data* yang terkirim ke *Stored Procedure* yang disimpan pada *database* salah, sehingga *data* yang tersimpan juga menjadi tidak tepat.

(d) Penggunaan *Stored Procedure* untuk validasi nomor Penjualan yang kurang tepat, jadi validasi yang digunakan untuk mengecek apakah no penjualan yang digunakan sudah pernah tercatat atau belum pada *database*, jika sudah maka tidak bisa disimpan, namun validasi ini tidak tepat sehingga nomor penjualan yang sama dapat dimasukkan ke dalam *database* terus menerus.

(e) Pembelian tidak mempengaruhi stok, jadi pada saat terjadi transaksi pembelian produk, stok yang menjadi tujuan produk itu disimpan tidak berubah.

(f) Pembelian tidak sesuai dengan gudang atau penyimpanan yang dipilih, jadi pada saat terjadi transaksi pembelian gudang atau tempat penyimpanan yang dipilih tidak tercatat sesuai dengan yang dipilih.

(g) *Form sale* tidak mau terbuka pada beda PC atau laptop, dikarenakan terjadi kesalahan *path* dari *crystal report* yang digunakan untuk membuat faktur penjualan.

(h) Penggunaan *Stored Procedure* yang belum berjalan pada *form adjustment* barang, jadi pada *form* ini seakan fitur yang di dalamnya tidak dapat digunakan dikarenakan perubahan penggunaan *databinding* yang ada.

(i) Fungsi dari *control* yang belum bekerja pada *form adjustment* barang, jadi tombol yang ada pada *form adjustment* ini belum berjalan dikarenakan memang semua *functionnya* sedang tidak dapat digunakan.

(j) Pada *form* pencarian barang yang terhubung pada *form adjustment* belum berjalan dengan benar.

(k) Tidak munculnya *list adjustment* pada *form list adjustment*, sama seperti *form adjustment, form list adjustment* juga belum berjalan dikarenakan *function* di dalamnnya belum benar.

(l) Tidak munculnya *list expedition* pada *form list expedition*, jadi pada *form list expedition* ini *function* yang digunakan untuk mengambil data ekspedisi belum ada.

(m) Penjualan melebihi stok yang tersedia pada *form Sale*, jadi pada saat terjadi transaksi penjualan produk yang terjual bisa melebihi stok sehingga stok menjadi minus.

(n) *Adjustment* yang belum mempengaruhi perubahan pada harga jual, jadi saat dilakukan penyesuai produk, harga baru yang ditentukan belum tersimpan dengan benar.

(o) Pada *form list sale* saat di*double click* tidak berbindah ke *form sale* dengan data yang dipilih, jadi seharusnya pada *list sale* pada saat di*double click* akan berpindah ke *form sale* namun semua data yang dipilih sudah berada pada *form sale* tersebut.

(p) Koneksi yang belum fleksibel, jadi koneksi pada *database* masih harus diubah secara manual atau di*hardcode*, dan setiap *form* perlu membuka koneksi sendiri – sendiri padahal pada saat aplikasi terbuka sudah membuka koneksi.

(q) Pada *form list purchase* jika di*double click* tidak menampilkan detailnya, serupa seperti *list adjustment* disini *function* yang digunakan untuk mengambil data daftar

penyesuaian masih belum ada dan ada sebagian yang kurang benar.

(r) Kesalahan dalam tempat pengambilan harga jual produk, jadi *table* awal yang digunakan berisi bukan daftar harga jual yang benar, jadi harus diubah diarahkan ke *table* yang benar.

(s) Validasi pada *form sale* yang masih kurang tepat, masih ada beberapa validasi yang digunakan untuk memastikan *input* dari *user* agar tidak salah yang masih kurang tepat.

(t) Pemilihan gudang tidak terpilih dengan benar pada *form sale*, pada saat terjadi transaksi penjualan gudang atau penyimpanan yang dipilih tidak sesuai dengan yang dikirim ke *database.*

(u) Kesalahan *query* pada faktur penjualan, terdapat sedikit kesalahan pada faktur penjualan dimana dalam *query* nya ada yang belum dipilih.

(v) *Form distribution product* tidak dapat memilih barang, jadi pada *form* ini *control* yang digunakan untuk memilih barang belum berjalan.

(w) *Form distribution product* tidak menunjukan jumlah barang dengan tepat sesuai dengan gudang yang dipilih, jadi pada saat pilihan gudang atau tempat penyimpanan berubah stok yang ditampilkan terkadang belum sesuai dengan aslinya.

(x) *Form payable transaction* belum dapat dijalankan, karena fungsi didalamnya belum dapat digunakan, pada *form* ini sebagian besar *function* yang diperlukan belum ada, jadi hanya susunan *control* pada *form* saja yang sudah ada.

(y) *Form receivable transaction* belum dapat dijalankan, karena fungsi didalamnya belum dapat digunakan, serupa dengan *payable transaction function* pada *form* ini juga sebagian besar belum ada.

(z) *Form search transaction* tidak berjalan karena belum ada *function* didalamnya yang digunakan untuk mencari transaksi yang ada.

(aa) Daftar stok produk tidak muncul, jadi pada *form* yang digunakan untuk melihat daftar stok per produk belum berjalan karena pada *query* serta *fucntion* nya terdapat kesalahan.

(bb) Daftar stok produk per gudang tidak muncul, serupa dengan daftar stok per produk, daftar stok per gudang atau penyimpanan produk juga memiliki sedikit kesalahan pada *query* serta *function* nya.

(cc) Tidak bisa menyimpan traksaksi penjualan jika *salesman* kosong, jadi pada saat menyimpan transaksi penjualan namun pada *dropdown salesman* kosong maka akan terjadi *crash*.

(dd) Tidak bisa menyimpan transaksi penjualan jika pelanggan kosong, pada saat terjadi transaksi penjualan, *dropdown customer* juga harus terisi jika tidak akan terjadi *crash*.

(ee) *Form Expedition* tidak berpengaruh pada stok saat disimpan, jadi pada saat *form expedition* diisi dan di *execute* jumlah barang yang dikirim dalam expedisi tidak mengurangi jumlah stok yang terdapat pada stok baik stok per barang maupun stok per gudang.

(ff) *Kesalahan query* laporan *profit and loss*, jadi terdapat sedikit kekurangan dan kesalahan pada kalkulasi yang digunakan pada pembuatan laporan *profit and loss.*

(gg) Pada daftar penjualan no transaksi belum terambil, jadi pada *form list sale* terdapat kesalahan *query* dan peletakaan data yang sudah diambil, sehingga no transaksi tidak muncul dalam *data grid view* yang telah disediakan.

(hh) Saat ingin melakukan *edit salesman* tidak bisa, terdapat kesalahan *function* pada *form entry salesman*,

(ii) Saat ingin melakukan *edit warehouse* tidak bisa, terdapat kesalahan *function* pada *form entry warehouse*.

(jj) Saat penjualan dan pembayaran dilakukan secara tunai, total pembayaran kurang dari total harga masih bisa, jadi seharusnya pada pembayaran tunai, seharusnya pembayaran akan langsung melunasi total harga jual yang ada baik itu pembayaran dengan uang pas atau lebih, tidak bisa kurang.

(kk) Kesalahan *query* laporan *inventory*.

(ll) *User* yang sama bisa login lagi dibeda tempat saat belum logout.

(mm) *User* biasa yang tidak memiliki kemampuna setara *admin* bisa membuka *setting User Login.*

Dari *bug* yang telah penulis jabarkan di atas, merupakan gambaran sebagian besar dari permasalahan *bug* yang ada pada *progrom Point Of Sales* yang penulis kerjakan dan berhasil penulis perbaiki selama penulis melakukan kegiatan magang pada *VBS Technology* ini.

Dari semua *bug* yang telah penulis jabarkan di atas, banyak dari sana yang tidak dapat penulis ambil gambarnya atau *screen shot*, selain karena memang ada yang penulis tidak ambil karena teralih oleh pekerjaan saat mengerjakannya sehingga penulis lupa, juga karena penulis tidak dapat terlalu banyak mengambil gambar dari *program Point Of Sales* ini karena tentu *program* ini merupakan milik dari perusahaan *VBS Technology*, sehingga tidak bisa sembarangan disebar kemanapun, berikut merupakan

beberapa contoh dari *screen shot* dari pekerjaan yang penulis kerjakan selama melakukan kegiatan magang di *VBS Technology*.

```
GetConfiguration()

ClassDB = New ClassDbConnection
ClassDB.SetConfiguration(Language, ServerName, DatabaseName, DbUserName, DbPassword)
ClassDB.GetConnections(xConn, xConn1, xConn2)

ObjFrmLogin.ShowDialog()
```

Gambar 3.3 Mendapatkan data untuk *connection*

Pada gambar di atas terlihat bagaimana pada sebuah *modul general* memanggil sebuah *function GetConfiguration*() yang kemudian dari data yang didapat dari sana dikirimkan kepada *classDB* yang kemudian digunakan untuk membuka koneksi kedalam *database.*

```
Public Sub GetConfiguration()
    If File.Exists(Application.StartupPath & "\Config.inf") Then
        tmpConfig = File.ReadAllLines(Application.StartupPath & "\Config.inf")
        For x As Integer = 0 To tmpConfig.Count - 1
            If tmpConfig(x).StartsWith("[Language]") Then
                Language = tmpConfig(x).Split("=")(1)
            ElseIf tmpConfig(x).StartsWith("[ServerName]") Then
                ServerName = tmpConfig(x).Split("=")(1)
            ElseIf tmpConfig(x).StartsWith("[DatabaseName]") Then
                DatabaseName = tmpConfig(x).Split("=")(1)
            ElseIf tmpConfig(x).StartsWith("[DbUserName]") Then
                DbUserName = tmpConfig(x).Split("=")(1)
            ElseIf tmpConfig(x).StartsWith("[DbPassword]") Then
                DbPassword = tmpConfig(x).Split("=")(1)
            End If
        Next
    End If
End Sub
```

Gambar 3.4 Mendapatkan data untuk *connection* dari file *external*

Pada gambar ini merupakan gambar dari *function GetConfiguration* yang dimana akan mengambil nilai dari *file config.inf* yang isinya merupakan data *Languate*, *ServerName*, *DatabaseName*, *DbUserName*, dan *DbPassword* yang akan digunakan dalam *connection string* untuk koneksi kedalam *database* yang terletak pada *server*.

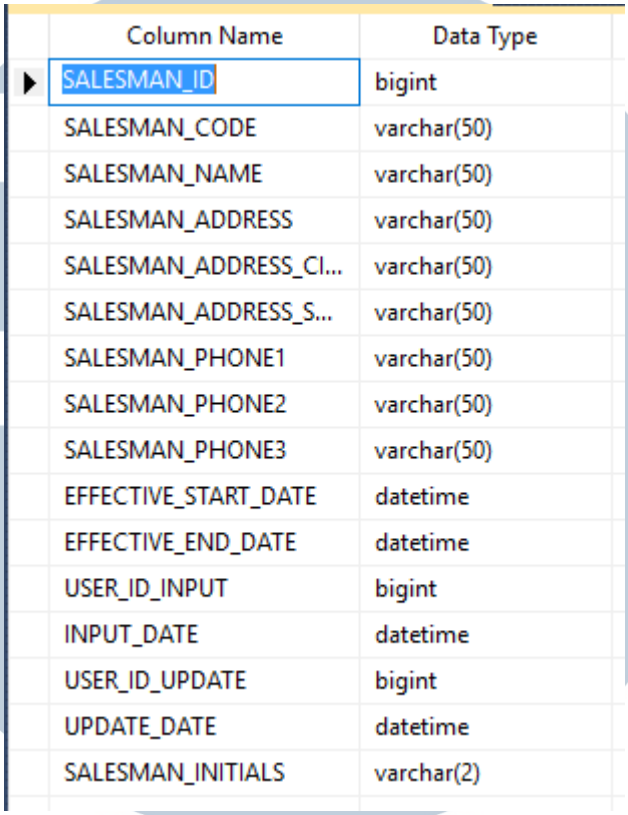
```
Public Sub SetConfiguration(ByVal pLanguage As String, ByVal pServerName As String, ByVal pDatabaseName As String,
                            ByVal pDbUserName As String, pDbPassword As String)
    Language = pLanguage
    ServerName = pServerName
    DatabaseName = pDatabaseName
    DbUserName = pDbUserName
    DbPassword = pDbPassword
End Sub

Public Sub GetConnections(ByRef pxConn As SqlConnection, ByRef pxConn1 As SqlConnection, ByRef pxConn2 As SqlConnection)
    OpenConnection()
    OpenConnection1()
    OpenConnection2()
    pxConn = xConn
    pxConn1 = xConn1
    pxConn2 = xConn2
End Sub

Public Sub OpenConnection()
    If xConn.State = ConnectionState.Open Then xConn.Close()
    xConn.ConnectionString = "Data Source=" & ServerName & ";Initial Catalog=" & DatabaseName & ";User Id=" &
        DbUserName & ";Password=" & DbPassword & ";MultipleActiveResultSets=true;"
    xConn.Open()
End Sub
```

Gambar 3.5 Isi dari *ClassDB*

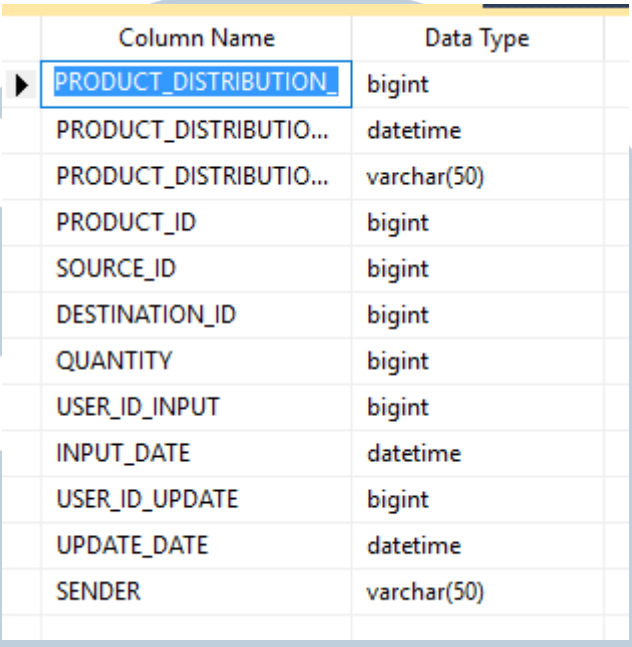
Gambar 3.5 berisi tentang isi dari *ClassDB* yang akan menjadi satu – satunya *class* yang digunakan untuk membuka koneksi kedalam *database* yang digunakan untuk *program Point Of Sales*, seperti yang terlihat pada Gambar 1 dimana setelah *GetConfiguration*(), nilai yang telah didapat dikirimkan kedalam *sub SetConfiguration* dalam *ClassDB* dan disana disimpan nilai yang dikirimkan dari *Modul General* tersebut, kemudian baru digunakan dalam *connectionString* untuk koneksi kedalam *database*, namun *connectionString* itu baru akan di*execute* pada saat *sub GetConnection* dipanggil balik oleh *modul* atau *class* yang membutuhkan koneksi kedalam database, maka dari itu *ClassDB* ini akan banyak di*import* pada banyak *form* atau *modul* lainnya yang sekiranya membutuhkan *class* ini.



Column Name	Data Type
SALESMAN_ID	bigint
SALESMAN_CODE	varchar(50)
SALESMAN_NAME	varchar(50)
SALESMAN_ADDRESS	varchar(50)
SALESMAN_ADDRESS_CI...	varchar(50)
SALESMAN_ADDRESS_S...	varchar(50)
SALESMAN_PHONE1	varchar(50)
SALESMAN_PHONE2	varchar(50)
SALESMAN_PHONE3	varchar(50)
EFFECTIVE_START_DATE	datetime
EFFECTIVE_END_DATE	datetime
USER_ID_INPUT	bigint
INPUT_DATE	datetime
USER_ID_UPDATE	bigint
UPDATE_DATE	datetime
SALESMAN_INITIALS	varchar(2)

Gambar 3.6 Struktur dari *table salesman*

Gambar 3.6 merupakan gambar dari struktur dari *table salesman* dimana penulis melakukan sedikit perubahan yaitu pada baris terakhir, yaitu *SALESMAN_INITIALS* yang digunakan untuk menyimpan inisial dari nama *salesman* yang dapat digunakan untuk memasukkan inisial itu kedalam faktur penjualan agar dapat mengetahui *sales* mana yang melakukan penjualan, dan untuk mengubah satu baris struktur dari *table salesman* ini, penulis perlu mengubah atau menambahkan beberapa hal, seperti menambahkan *field* pada *entry salesman* untuk meletakkan data inisial tersebut, dan mengubah *Stored Procedure* serta *query* yang digunakan agar sesuai dengan *table salesman* tersebut, juga *query* yang digunakan pada faktur penjualan juga perlu diubah, karena perlu memasukkan inisial didalamnya.



Column Name	Data Type
PRODUCT_DISTRIBUTION_	bigint
PRODUCT_DISTRIBUTIO...	datetime
PRODUCT_DISTRIBUTIO...	varchar(50)
PRODUCT_ID	bigint
SOURCE_ID	bigint
DESTINATION_ID	bigint
QUANTITY	bigint
USER_ID_INPUT	bigint
INPUT_DATE	datetime
USER_ID_UPDATE	bigint
UPDATE_DATE	datetime
SENDER	varchar(50)

Gambar 3.7 Struktur *table product distribution*

Pada gambar 3.7 ditunjukkan struktur *table product distribution* yang telah ditambahkan kolom baru, yaitu *SENDER* yang sebelumnya telah disebutkan juga di atas, dan untuk menambahkan satu kolom ini juga ada beberapa hal yang perlu diubah, yaitu seperti *control* pada *form product distribution* itu sendiri, serta *Stored Procedure* yang digunakan agar sesuai dengan semua *field* yang tersedia dan dibutuhkan pada *table product distribution* itu sendiri, kolom *SENDER* ini akan digunakan untuk menyimpan data siapakah yang melakukan sebuah pengiriman atau distribusi dari suatu produk, yang nantinya namanya akan dicantumkan kedalam surat jalan yang dapat dicetak juga, sehingga *query* yang digunakan untuk membuat *report* surat jalan menggunakan *crystal report* pun harus diubah juga.

Column Name	Data Type
PRODUCT_ID	bigint
PRICE	bigint
EFFECTIVE_START_DATE	date
EFFECTIVE_END_DATE	date
USER_ID_INPUT	bigint
INPUT_DATE	date
USER_ID_UPDATE	bigint
UPDATE_DATE	date

Gambar 3.8 Struktur dari *table product purchase price*

Pada gambar 3.8 berisi struktur dari *table product purchase price* yang penulis buat untuk menampun harga dari suatu produk pada saat dibeli untuk mengisi stok dari produk yang dibeli tersebut, tujuan dibuat dari *table* ini adalah untuk digunakan pada saat perhitungan *profit and loss* dari *client* yang menggunakan program ini, namun *PRICE* pada *table* di atas berisi rata – rata dari harga produk yang dibeli agar mempermudah kalkulasi, dan perubahan dari *PRICE* tersebut tidak diisi manual, melainkan menggunakan *trigger* yang penulis buat juga sebelumnnya, sehingga setiap terjadi pembelian maka *table* ini akan secara otomatis berubah sesuai dengan produk yang dibeli.

Column Name	Data Type
PRODUCT_ID	bigint
PRODUCT_CODE	varchar(50)
PRODUCT_NAME	varchar(50)
DESCRIPTION	varchar(60)
SELLING_PRICE	bigint
LOCATION_ID	bigint
SAFE_STOCK	bigint
USER_ID_INPUT	bigint
INPUT_DATE	datetime
USER_ID_UPDATE	bigint
UPDATE_DATE	datetime
CATEGORY	int
EFFECTIVE_START_DATE	datetime
EFFECTIVE_END_DATE	datetime
PRODUCT_GROUP_ID	bigint
PRODUCT_TYPE_ID	bigint

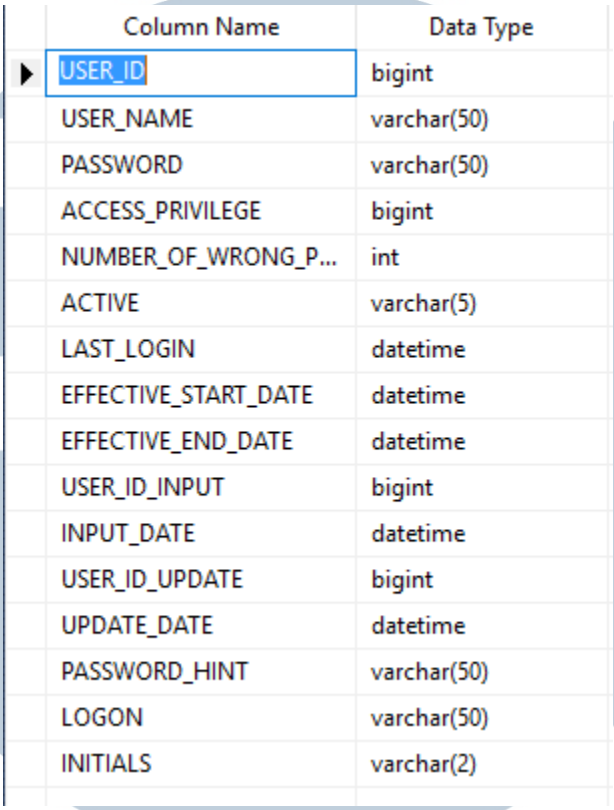
Gambar 3.9 Struktur *table product*

Gambar 3.9 berisikan struktur dari *table product* yang isinya telah diubah sedikit juga, yaitu pada baris terakhir yaitu *PRODUCT TYPE ID*, yang berisikan no *ID* dari produk tersebut yang mengidentifikasikan termasuk tipe apakah produk tersebut, seperti biasa dalam menambahkan satu kolom tersebut terdapat beberapa hal yang perlu penulis ubah, yaitu seperti menambahkan *field* untuk menampung *PRODUCT TYPE ID* pada *form entry product*, serta mengubah *Stored Procedure* agar sesuai dengan perubahan yang telah dibuat pada pada *table product* ini juga.

Column Name	Data Type	Allow Nulls
PRODUCT_TYPE_ID	bigint	☐
PRODUCT_TYPE_CODE	varchar(50)	☐
PRODUCT_TYPE_NAME	varchar(50)	☐
EFFECTIVE_START_DATE	date	☑
EFFECTIVE_END_DATE	date	☑
USER_ID_INPUT	bigint	☐
INPUT_DATE	date	☐
USER_ID_UPDATE	bigint	☑
UPDATE_DATE	date	☑

Gambar 3.10 Struktur dari *table product type*

Gambar 3.10 merupakan berisikan struktur *table product type* yang penulis buat untuk menampung tipe dari produk yang ada, tujuannya adalah agar setiap dari produk memiliki tipe – tipe tersendiri agar lebih mudah dalam pengelompokkannya, misalnya saja ada produk baju, kaos akan masuk ke dalam tipe pakaian, untuk memenuhi tujuan yang telah disebutkan tersebut, selain diperlukan membuat *table,* diperlukan juga sebuah *form* yang dapat digunakan untuk memasukkan tipe produk apa saja yang sekiranya dibutuhkan oleh *client* itu sendiri, dan selain dari *form,* dibutuhkan juga *Stored Procedure* untuk melakukan *INSERT, UPDATE,* dan *DELETE* pada *table product type* ini.



Column Name	Data Type
USER_ID	bigint
USER_NAME	varchar(50)
PASSWORD	varchar(50)
ACCESS_PRIVILEGE	bigint
NUMBER_OF_WRONG_P...	int
ACTIVE	varchar(5)
LAST_LOGIN	datetime
EFFECTIVE_START_DATE	datetime
EFFECTIVE_END_DATE	datetime
USER_ID_INPUT	bigint
INPUT_DATE	datetime
USER_ID_UPDATE	bigint
UPDATE_DATE	datetime
PASSWORD_HINT	varchar(50)
LOGON	varchar(50)
INITIALS	varchar(2)

Gambar 3.11 Struktur *table user*

Gambar 3.11 berisikan tentang struktur dari *table user* yang juga telah dilakukan perubahan, yaitu pada dua baris terakhir yaitu *LOGON* dan *INITIALS*, dimana *LOGON* digunakan untuk menyimpan status *login* dari sebuah *user* yang ada, dan *LOGON* ini bertujuan untuk mencegah *user* yang sama untuk dapat melakukan *login* dilebih dari satu tempat secara bersamaan kecuali *admin*, karena *admin* akan berperan untuk dapat mengubah status *login* dari suatu *user* jika suatu waktu akun yang mereka gunakan tersangkut dengan status *login* terus.

Kemudian terdapat *INITIALS* yang digunakan untuk mencatat inisial dari *user* yang ada, hal ini bertujuan serupa dengan kolom *INITIALS* pada *salesman*, dan ini bertujuan agar menghindari *user* yang berbeda melakukan transaksi yang sama, jadi setiap transaksi penjualan akan tercantum inisital dari *user* yang melakukan transaksi tersebut.

Untuk menambahkan dua kolom tersebut, penulis perlu menambahkan satu *field* baru pada *entry user* yang digunakan untuk

menampung *INITIALS*, serta melakukan sediki perubahan pada *Stored Procedure* yang digunakan agar sesuai dengan perubahan yang telah dibuat pada *table user* ini.

```
USE [DB_DMI_RETAIL]
GO
/****** Object:  StoredProcedure [dbo].[SP_CHECK_USED_PRODUCT_TYPE]    Script Date: 12/5/2016 8:04:00 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <David Chandra>
-- Create date: <25-08-2016>
-- Description:
-- =============================================
ALTER PROCEDURE [dbo].[SP_CHECK_USED_PRODUCT_TYPE]
    -- Add the parameters for the stored procedure here
    @PRODUCT_TYPE_CODE VARCHAR(50) ,
    @RESULT int = 0 OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT @RESULT = (SELECT COUNT(*) FROM PRODUCT WHERE PRODUCT_TYPE_ID =
                      (SELECT PRODUCT_TYPE_ID FROM PRODUCT_TYPE WHERE PRODUCT_TYPE_CODE = @PRODUCT_TYPE_CODE))
END
```

Gambar 3.12 *Stored Procedure*

Pada gambar 3.12 ini mengenai *Stored Procedure* yang penulis buat pada saat melakukan kegiatan magang, *Stored Procedure SP CHECK USED PRODUCT TYPE* ini dibuat bertujuan untuk mengecek apakah tipe produk yang dipilih digunakan oleh produk atau dengan kata lain apakah terdapat produk yang menggunakan produk tipe tersebut, hal ini bertujuan agar pada saat suatu tipe produk hendak dihapus tidak ada data lain yang menggunakan data yang hendak dihapus ini, sehingga tidak akan terjadi *error* atau *null* pada saat data tersebut dihapus dari *table product type* tersebut.

```
USE [DB_DMI_RETAIL]
GO
/****** Object:  StoredProcedure [dbo].[SP_DELIVERY_ORDER_LIST]    Script Date: 12/5/2016 8:08:48 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      David Chandra
-- Create date: <26/7/2016>
-- Description: <List Delivery Order>
-- =============================================
ALTER PROCEDURE [dbo].[SP_DELIVERY_ORDER_LIST]
    -- Add the parameters for the stored procedure here
    @PERIOD1 DATE,
    @PERIOD2 DATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT DELIVERY_ORDER_NO "Delivery Number",
        CAST(DALIVERY_DATE AS DATE)"Delivery Date",
        (SELECT CUSTOMER_NAME FROM CUSTOMER WHERE CUSTOMER_ID = DELIVERY_ORDER_HEADER.CUSTOMER_ID)"Customer Name",
        DELIVERY_ADDRESS "Delivery Address",
        (SELECT EXPEDITION_NAME FROM EXPEDITION WHERE EXPEDITION_ID = DELIVERY_ORDER_HEADER.EXPEDITION_ID)"Expedition Name"
    FROM DELIVERY_ORDER_HEADER
    WHERE CAST(DALIVERY_DATE AS DATE) BETWEEN @PERIOD1 AND @PERIOD2
        AND DELIVERY_ORDER_ID NOT IN (SELECT DELIVERY_ORDER_ID WHERE VOID = 'TRUE')
    ORDER BY DALIVERY_DATE
END
```

Gambar 3.13 ***Stored Procedure delivery order list***

Pada gambar 3.13 ini, menampilkan sebuah *Stored Procedure* yang digunakan untuk mengambil data dari *table delivery order* yang kemudian akan di tampilkan pada *form delivery order list*, *Stored Procedure* ini akan menampilkan data *delivery order* yang tanggal pengirimannya berada diantara dua tanggal yang sebelumnya telah ditentukan terlebih dahulu melalui *form delivery order* dan juga dimana status *void* pada *table delivery order* tidak *TRUE*, dimana maksudnya adalah pada saat pengiriman dibatalkan atau tidak jadi dilakukan maka *order* tersebut dapat dijadikan *void* dengan mengubahnya menjadi *TRUE*, namun tidak menghabus data pada *table* tersebut agar tetap terdapat *history* dari *delivery order* tersebut walaupun telah *void.*

```
USE [DB_DMI_RETAIL]
GO
/****** Object:  StoredProcedure [dbo].[SP_GENERATE_PRODUCT_TYPE_CODE]    Script Date: 12/5/2016 8:02:47 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <David Chandra>
-- Create date: <25-08-2016>
-- Description:
-- =============================================
ALTER PROCEDURE [dbo].[SP_GENERATE_PRODUCT_TYPE_CODE]
    -- Add the parameters for the stored procedure here
    @NUMBER VARCHAR(50) OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    IF EXISTS (SELECT * FROM PRODUCT_TYPE)
    BEGIN
        SELECT @NUMBER = (SELECT 'PRT-' +CAST(MAX(PRODUCT_TYPE_ID) + 1 As Varchar(50))FROM PRODUCT_TYPE)
    END
    ELSE
    BEGIN
        SELECT @NUMBER = (SELECT 'PRT-1')
    END
END
```

Gambar 3.14 ***Stored Procedure Generate Product Type Code***

Pada gambar 3.14 ini menampilkan *Stored Procedure Generate Product Type Code* yang penulsi buat, yang bertujuan untuk membuat kode untuk tipe produk secara otomatis untuk menghindari duplikasi kode tipe produk yang dimasukkan oleh *user*, pada *Stored Procedure* ini, pertama – tama akan mengecek apakah sudah ada data pada *table product type* atau belum, jika belum ada maka *Stored Procedure* ini akan langsung mengembalikan nilai PRT-1, sedangkan jika sudah ada data di dalam *table product type* tersebut maka *Stored Procedure* akan menggabungkan PRT- dengen nilai maksimal atau terbesar dari *Product Type ID*, yang kemudian ditambah satu, baru setelah itu akan dikirimkan kembali nilai dari kode tipe produk ini ke *program Point Of Sales* ini untuk dilanjutkan ke tahap berikutnya.

```
USE [DB_DMI_RETAIL]
GO
/****** Object:  StoredProcedure [dbo].[SP_PRODUCT_TYPE]    Script Date: 12/5/2016 8:04:59 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <David Chandra>
-- Create date: <25-08-2016>
-- Description: <Description,,>
-- =============================================
ALTER PROCEDURE [dbo].[SP_PRODUCT_TYPE]
    -- Add the parameters for the stored procedure here
    @METHOD CHAR(1),
    @PRODUCT_TYPE_ID BIGINT,
    @PRODUCT_TYPE_CODE VARCHAR(50),
    @PRODUCT_TYPE_NAME VARCHAR(50),
    @EFFECTIVE_START_DATE DATETIME,
    @EFFECTIVE_END_DATE DATETIME,
    @USER_ID_INPUT BIGINT,
    @INPUT_DATE DATETIME,
    @USER_ID_UPDATE BIGINT,
    @UPDATE_DATE DATETIME
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    BEGIN
    IF @METHOD = 'I'
    BEGIN
        IF EXISTS (SELECT PRODUCT_TYPE_ID FROM PRODUCT_TYPE)
        BEGIN
            SELECT @PRODUCT_TYPE_ID = (SELECT MAX(PRODUCT_TYPE_ID) FROM PRODUCT_TYPE) + 1
        END
        ELSE
        BEGIN
            SELECT @PRODUCT_TYPE_ID = 1
```

Gambar 3.15 ***Stored Proceduer Product Type part 1***

```
        INSERT INTO PRODUCT_TYPE(PRODUCT_TYPE_ID,
                                 PRODUCT_TYPE_CODE,
                                 PRODUCT_TYPE_NAME,
                                 EFFECTIVE_START_DATE,
                                 EFFECTIVE_END_DATE,
                                 USER_ID_INPUT,
                                 INPUT_DATE,
                                 USER_ID_UPDATE,
                                 UPDATE_DATE)
                         VALUES(@PRODUCT_TYPE_ID,
                                @PRODUCT_TYPE_CODE,
                                @PRODUCT_TYPE_NAME,
                                @EFFECTIVE_START_DATE,
                                @EFFECTIVE_END_DATE,
                                @USER_ID_INPUT,
                                @INPUT_DATE,
                                @USER_ID_UPDATE,
                                @UPDATE_DATE)
    END

    IF @METHOD = 'U'
    BEGIN
        UPDATE PRODUCT_TYPE
        SET PRODUCT_TYPE_CODE = @PRODUCT_TYPE_CODE,
            PRODUCT_TYPE_NAME = @PRODUCT_TYPE_NAME,
            EFFECTIVE_START_DATE = @EFFECTIVE_START_DATE,
            EFFECTIVE_END_DATE = @EFFECTIVE_END_DATE,
            USER_ID_INPUT = @USER_ID_INPUT,
            INPUT_DATE = @INPUT_DATE,
            USER_ID_UPDATE = @USER_ID_UPDATE,
            UPDATE_DATE = @UPDATE_DATE
        WHERE PRODUCT_TYPE_ID = @PRODUCT_TYPE_ID
    END

    IF @METHOD = 'D'
    BEGIN
        DELETE PRODUCT_TYPE WHERE PRODUCT_TYPE_ID = @PRODUCT_TYPE_ID
    END
END
END
```

Gambar 3.16 ***Stored Proceduer Product Type Part 2***

Pada gambar 3.15 dan gambar 3.16 merupakan *Stored Procedure Product Type* yang digunakan untuk keperluan *INSERT, UPDATE,* dan *DELETE* untuk *table product type,* jadi *Stored Procedure* ini yang akan bekerja setiap setiap akan terjadi perubahan pada *table product type* ini melalui *form entry product type,* jadi cara kerja dari *Stored Procedure* ini adalah pada awalnya akan menerima masukkan data dari *program Point Of Sales,* dari data – data yang masuk akan terdapat *method* yang akan berguna untuk menentukan *query* apakah yang akan dijalankan, antara *INSERT, UPDATE,* ataukan *Delete,* setelah itu baru data – data yang lainnya akan masuk ke dalam *query* yang bersangkutan atau yang sesuai dengan pilihan *method* yang ada, jika *insert* maka akan menjalankan *query insert* seperti biasa, jika *method* yang dipilih adalah *update,* maka akan melakukan *update* isi data yang sudah ada pada *table,* yang disesuaikan berdasarkan dari *product type id* yang ada, dan yang terakhir adalah *delete*, dimana akan menjalanakn *query delete* biasa namun hanya *product type* dengan *product type id* tertentu saja yang yang akan dihapus, tidak semuanya.

```
Private Sub SP_PRODUCT_TYPE(ByVal pMETHOD As String, ByVal pPRODUCTTYPEID As Integer, ByVal pPRODUCTTYPECODE As String,
                            ByVal pPRODUCTYPTENAME As String, ByVal pEFFECTSTART As DateTime, ByVal pEFFECTEND As DateTime, ByVal pUSER_ID_INPUT As Integer,
                            ByVal pINPUT_DATE As DateTime, ByVal pUSER_ID_UPDATE As Integer, ByVal pUPDATE_DATE As DateTime)
    If xConn.State = ConnectionState.Open Then xConn.Close() : xConn.Open()

    xComm = New SqlCommand
    xComm.Connection = xConn
    xComm.CommandType = CommandType.StoredProcedure
    xComm.CommandText = "SP_PRODUCT_TYPE"
    xComm.Parameters.AddWithValue("@METHOD", pMETHOD)
    xComm.Parameters.AddWithValue("@PRODUCT_TYPE_ID", pPRODUCTTYPEID)
    xComm.Parameters.AddWithValue("@PRODUCT_TYPE_CODE", pPRODUCTTYPECODE)
    xComm.Parameters.AddWithValue("@PRODUCT_TYPE_NAME", pPRODUCTYPTENAME)
    xComm.Parameters.AddWithValue("@EFFECTIVE_START_DATE", pEFFECTSTART)
    xComm.Parameters.AddWithValue("@EFFECTIVE_END_DATE", pEFFECTEND)
    xComm.Parameters.AddWithValue("@USER_ID_INPUT", pUSER_ID_INPUT)
    xComm.Parameters.AddWithValue("@INPUT_DATE", pINPUT_DATE)
    xComm.Parameters.AddWithValue("@USER_ID_UPDATE", pUSER_ID_UPDATE)
    xComm.Parameters.AddWithValue("@UPDATE_DATE", pUPDATE_DATE)
    xComm.ExecuteNonQuery()
End Sub
```

Gambar 3.17 *sub product type*

Gambar 3.17 ini merupakan *sub SP PRODUCT TYPE* yang berada pada *program Point Of Sales, sub* ini digunakan untuk mengirimkan nilai yang telah terkumpul pada *form product type* sesuai dengan kriterianya menuju ke *Stored Procedure* yang terletak pada *database server*, sehingga semua kalkulasi dilakukan di *server*, agar tidak terlalu membebani *program* serta *PC* maupun laptop yang digunakan oleh *client.*

3.4 Pekerjaan Tambahan yang Dikerjakan

Selama melakukan pekerjaan utama yang penulis dapatkan dari *supervisor* sekaligus pemilik dari *VBS Technology* ini, penulis juga mendapatkan pekerjaan tambahan yang perlu penulis kerjakan, antara lain adalah

(a) Menambahkan kolom pada *table user*, jadi untuk menyelesaikan suatu masalah atau *bug* yang terjadi pada *program Point Of Sales* ini, *supervisor* penulis mendapatkan sebuah solusi untuk mengatasinya, dan solusi itu memerlukan penulis untuk menambahakan kolom pada *table user* yang telah ada.

(b) Membuat atau agar setiap faktur pejualan yang ada akan memiliki sebuah kode yang berisi *user* yang menggunakan *program Point Of Sales* ini, serta beberapa data lain termasuk *counter* agar setiap kode unik antara satu sama lain, juga unik di setiap bulan, karena *counter* tersebut akan di*reset* setiap bulannya.

(c) Membuat *table* baru yang berguna untuk menampung tipe dari produk yang ada, jadi *table* yang baru ini akan dapat mengelompokkan produk – produk berdasarkan tipenya, bukan hanya tipe produk atau jasa, tapi lebih ke arah jenis dari produk itu apa, dan untuk membuat ini juga termasuk penulis perlu dalam melakukan sedikit perubahan pada *table product* sehingga dapat menampung nilai dari tipe produk yang sebelumnya telah di*set* atau *input* juga oleh *user*, dan tentu agar *user* dapat melakukan *input* sendiri, maka diperlukan sebuah *form* baru yang digunakan untuk menangkap inputan dari user, dan tentu ini juga berhubungan dengan *function* yang diperlukan di dalamnya dimana *function* tersebut akan membutuhkan sebuah *stored procedure* baik untuk melakukan *INSERT,UPDATE,* atau *DELETE* pada *table tipe* produk yang telah penulis buat.

(d) Membuat *table* baru lainnya yang digunakan untuk menampung harga beli suatu produk, dan maka dari itu penulis perlu melakukan sedikit perubahan pada *form* pembelian produk, dimana akan terdapat sebuah *function* yang akan membuat data dari *form* pembelian tersebut akan ada yang masuk juga ke dalam, dan penulis juga diminta untuk membuat sebuah *trigger* yang akan berfungsi agar pada saat terjadi pembelian maka *table* yang digunakan untuk menampung harga beli ini akan secara otomatis diperbaharui sesuai dengan rata – ratanya, karena pada *program Point Of Sales* ini menggunakan teknik *average* dalam menghitung perhitungan *profit and loss.*

(e) Membuat sebuah *setting* pada menu *setting* yang dapat digunakan untuk mengunci apakah nomor transaksi akan di*generate* secara manual atau otomatis, dan untuk membuatnya perlu ditambahkan *control* baru berupa sebuah *check box*, selain itu perlu ditambahkannya sebuah kolom baru pada *table* yang menyimpan *setting*, juga perlunya ditambahkan sebuah *query* baru pada *form setting* tersebut, juga beserta dengan sebuah validasi untuk mengecek apakah *check box* tersebut dicentang atau tidak, yang nilai dari validasi tersebut yang akan dimasukkan ke dalam *query* tersebut.

(f) Memasukkan data – data asli milik dari *client* yang akan digunakan tidak hanya untuk uji coba secara nyata *program Point Of Sales* ini, tapi juga agar pada saat *program* ini diberikan kepada *client* maka *program* ini sudah siap digunakan oleh mereka.

(g) Membuat *control* baru pada menu distirbusi produk yang berfungsi untuk menyimpan sekaligus mencetak surat jalan mengenai distribusi produk yang baru terjadi, dalam membuatnya penulis perlu membuat sebuah laporan atau *report* baru yang dapat menampung sekiranya seluruh *data* yang

diperlukan dalam surat jalan tersebut, penulis juga perlu menambahkan sebuah kolom baru pada *table* distribusi barang, untuk membantu melengkapi data yang dibutuhkan dalam surat jalan, juga membuat agar dari menu *list distribution produk* jika suatu data dipilih dan di*double click* maka, akan dapat masuk ke *entry distribution product*, dengan menampilkna detailnya dan dapat mencetak kembali surat jalan yang ada.

3.5 Masalah yang Dihadapi

Selama melakukan kegiatan magang pada *VBS Technology* ini tentu terdapat berbagai kendalam yang dihadapi penulis, dan kendala – kendala atau masalah ini lah yang membuat penulis mendapatkan pengalam – pengalaman dalam mengatasi kesulitan dalam dunia kerja nyata yang akan penulis hadapi juga dikemudian hari, berikut adalah kendala – kendala utama yang penulis temui selama masa magang pada *VBS Technology* ini,

1. Pada *program* yang penulis cari *bug* nya itu sudah jadi sebenarnya namun karena berbagai macam hal maka jadi terdapat banyak *bug* dan karena *program* itu sudah jadi penulis harus memahami terlebih dahulu alur *program* tersebut dimana tim yang dulu membuat program tersebut sudah tidak ada sehingga penulis mau tidak mau harus mencari tahu hal itu sendiri, ditambah juga karena tim yang sebelumnya tentu terdiri tidak hanya dari satu orang, maka terkadang pada setiap *form* memiliki logika yang berbeda – beda sehingga setiap mengerjakan *form* berbeda tidak jarang penulis perlu mencari tahu lagi dari awal maksud dan alur dari logika pemgrograman yang ada.
2. *Database* yang digunakan juga berbeda dengan yang pernah penulis gunakan atau pelajari sehingga penulis harus belajar hal itu terlebih dahulu, walau sebenarnya tidak terlalu berbeda dengan yang pernah penulis pelajari, yang paling berbeda adalah tampilannya, namun karena pada *database* yang penulis gunakan ini terhubung langsung dengan *server* milik *VBS Technology* ini, tentu dalam pemakaiannya harus berhati – hati karena

perubahan di dalamnnya akan terpengaruh ke semua orang yang menggunakan *database* itu juga.

3. Terdapat beberapa *query* yang digunakan untuk mengambil atau memasukkan *data* ke *database* terkadang ada yang menggunakan *Stored Procedure* dimana penulis juga belom mempelajarinya sehingga pada awal melakukan magang penulis merasa kebingungan dengan hal itu.
4. Pada saat pembuatan laporan di*program* ini, cara pembuatannya juga berbeda dengan yang dulu pernah penulis kerjakan, dimana hal ini juga mempersulit penulis, laporan yang penulis maksud disini adalah berupa pembuatan *report* menggunakan *crystal report* namun pada pengambilan datanya penulis tidak langsung mengambil dari keseluruhan dari satu *table* yang ada, melainkan penulis perlu membuat *query select* terlebih dahulu memilih data – data dari berbagai *table* yang sekiranya diperlukan untuk dimasukkan ke dalam *report* yang sedang penulis buat.

3.6 Solusi Atas Masalah

Dalam penyelesaian tugas yang penulis dapat pada pelaksanaan kerja magang ini yaitu melakuakn *Debug* dan penambahan fitur yang dibutuhkan *client*, penulis telah mengerjakan atau memperbaiki fitur - fitur utamanya, dan menambah beberapa fitur tambahan sesuai kebutuhan, sehingga *program Point of Sales* ini sudah dapat digunakan untuk kegiatan *client* seperti sebagaimana seharusnya, walaupun belum semua fitur yang sebenarnya sudah ada pada program *Point of Sales* ini sudah dapat digunakan karena masih terdapat banyak *bug* yang perlu diperbaiki sebelum program *Point of Sales* ini dapat digunakan seluruhnya.

Mengenai masalah yang dihadapi dalam *proses* pelaksanaan magang ini, dalam menyelesaikannya atau mengatasinya penulis melakukan beberapa hal, yaitu

1. Pertama mengenai alur program, penulis memahaminya perlahan dengan membaca *coding* yang ada dan mengukuti jalannya setahap demi setahap sampai akhirnya penulis mulai mengerti, karena jika penulis tidak bisa mengerti hal itu maka tidak akan bisa memperbaiki *bug* yang ada, dan

terkadang tetap terdapat hal yang penulis kurang mengerti, dan penulis perlu bertanya kepada *supervisor* untuk membantu menejlaskan maksudnya, juga karena terkadang *supervisor* penulis tidak ada karena urusan pekerjaan juga maka penulis perlu mencari sendiri menggunakan *google*.

2. Kedua yaitu mengenai *database*, ditempat penulis melakukan magang ini, database yang digunakan adalah *SQL*, sebenarnya penulis telah mempelajari itu namun yang berbeda adalah yang digunakan untuk mengatur database tersebut, pada tempat magang penulis menggunakan *SQL Server Management Studio* dimana penulis baru kali ini menggunakannya, pada masalah ini, selalin penulis mencoba - coba menggunakannya penulis juga bertanya kepada teman magang penulis yang pernah menggunakannya agar menjelaskan cara penggunaanya agar penulis tidak membuat kesalahan pada saat melakukan perubahan atau penambahan pada sebuah *database* karena itu akan berakibat langsung kepada seluruh orang yang terkoneksi langsung ke *server*, dan penulis juga pernah mendapatkan contoh beberapa cara penggunaan dari *SQL Server Management Studio* ini dari *supervisor* penulis.
3. Ketiga mengenai *Stored Procedure*, hal ini merupakan benar-benar yang pertama yang penulis temui, dan untuk mengatasinya penulis mencari di*forum* mengenai hal itu, serta melihat *Stoder Procedure* yang sudah ada dan digunakan pada program ini sehingga penulis dapat mempelajarinya dan menirunya untuk membuat *Stored Procedure* lainnya yang penulis butuhkan, namun sebelum penulis memasukkan *Stored Procedure* yang penulis buat, penulis perlu melakukan *test* terlebih dahulu apakah kira – kira *Stored Procedure* yang penulis buat sudah benar atau belum.
4. Keempat mengenai pembuatan laporan yang akan digunakan oleh *Client* penulis pada awal mencari - cari di*forum* mengenai hal ini, dan setelah itu penulis juga diberi contoh oleh *supervisor* penulis dalam cara pembuatannya, sehingga penulis pun dapat membuat laporan yang dibutuhkan oleh *client*.