# Accela Database Reporting Basics

## Report query writing in a Self-Hosted environment

Erin Griffith

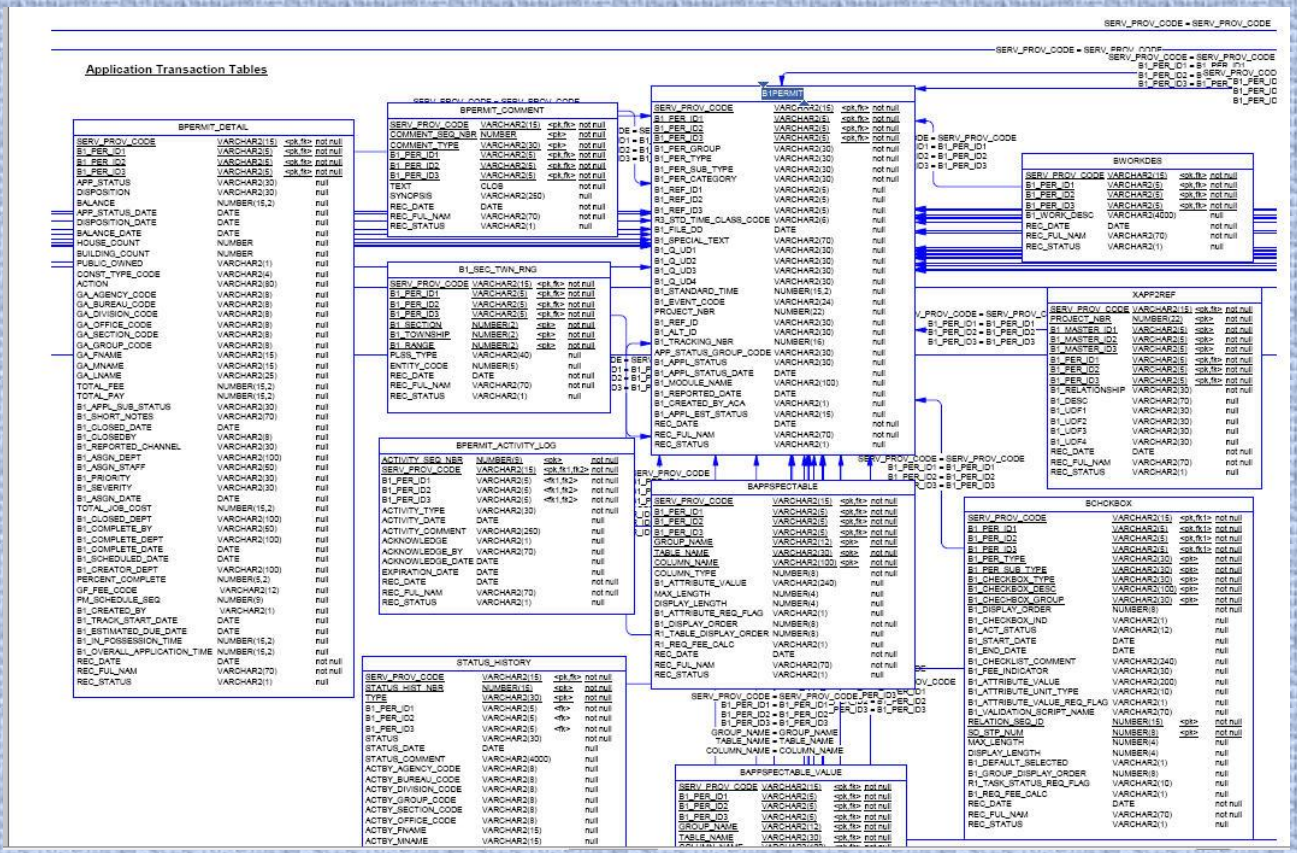
May 2012

# Database Overview

- Accela Database
- Data Dictionary
- Table Notes
- Important Transaction Tables
- Data Dictionary ERD
- Accela Functions

# Accela Database

There are approximately 725 tables (version 7.1)



# Data Dictionary

"Data Dictionary of Common Accela Automation.pdf"

## A. Daily Activities – CAP Related Data

*This section outlines the table structure for all data relating to an individual record*

**B1PERMIT table** Database Table Name

| AA Field | Column Name | Comment |
|---|---|---|
| Record ID / Permit Number / Case Number / CAP Number | B1_ALT_ID<br>Database Column Name | |

## B. Administrator Tools – Configuration and Reference Data

*These are tables not related to CAP records (Staff information, Application Type, Look-Ups, etc.)*

## C. Appendices

*This section gives tips on locating Accela fields in the Data Dictionary*

## B. Entity Relationship Diagrams - ERD

*This section shows how certain tables relate and what fields are used to join them*

***Very useful when reporting on financial information***

***<u>STUDY THIS DOCUMENT!!!</u>***

# Data Dictionary – SQL Side

There are two tables in the Accela Database that act as a data dictionary and will give you information on the tables and their fields. AA_DATA_DIC and AA_OBJECTS.

Here are some example queries:

- SELECT * FROM AA_DATA_DIC WHERE COLUMN_NAME LIKE '%ADDRESS%'
- SELECT * FROM AA_OBJECTS WHERE OBJECT_NAME LIKE '%TIME%'

# Table Notes

The Accela database is made up of two types of tables:

- Transactional

  Example: B1Permit, B3Address, B3Parcel, F4Payment, F4Feeitem
  These are tables that hold transactions relating to a permit.

- Reference

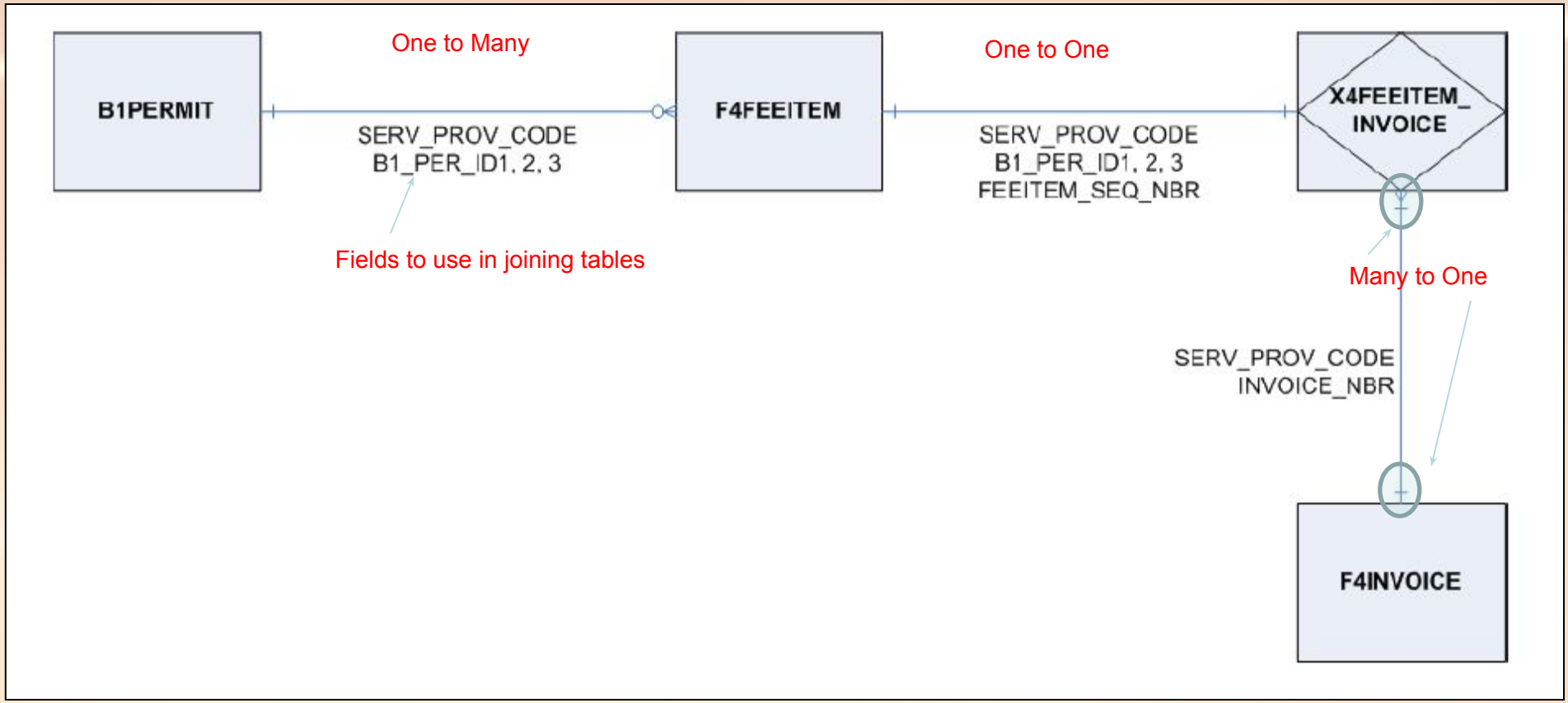
  Example: R2chckbox, R3Apptyp, SProcess, X4Payment_Feeitem
  These tables hold information relating to the setup, configuration or Cross Reference (between tables)

# Important Transaction Tables

- BCHCKBOX
  - Holds ASI information
  - Data is accessed for reports using a function

    dbo.FN_GET_APP_SPEC_INFO(B.SERV_PROV_CODE,B.B1_PER_ID1,B.B1_PER_ID2,B.B1_PER_ID3,‘ASI Field')
- BAPPSPECTABLE_VALUE
  - Holds ASIT information
  - Each column in the ASIT table is stored in a row in this table
  - Data can be accessed using a function or by creating a query that normalizes the data
- B1PERMIT
  - Main CAP record table
  - Along with BPERMIT_DETAIL it stores most record level information (fields that only have one per CAP record)

# Data Dictionary ERD



# Accela Functions

”AA Oracle Database Function Reference Guide.pdf”

- Lists all functions maintained by Accela and the parameter fields required to execute them
- Functions should ALWAYS be used when possible for efficiency
- Commonly Used Functions
  - FN_GET_APP_SPECIFIC_INFO: Used to pull ASI data
  - FN_GET_PRI_ADDRESS_FULL: Returns full primary address in multi-line format
  - FN_GET_PRI_ADDRESS_PARTIAL: Returns the first line of the address
  - FN_GET_CONTACT_INFO: Returns various contact level information
  - FN_GET_OWNER_INFO: Returns various owner level information
  - FN_GET_PARCEL_NBR: Returns the parcel number

# SQL Overview

- Very Basic SQL Syntax
- Basic SQL Syntax
- Summary Syntax
- Summary Syntax with Accela
- Accela Query Reminders
- Using Accela Functions
- Useful SQL Functions

# Very Basic SQL Syntax

```
SELECT   *
FROM     B1PERMIT
WHERE    SERV_PROV_CODE = '[Insert Client ID]'
```

- SELECT
  - Section where fields to be returned as columns are designated
  - Separate fields with commas
  - An “*” can be used to return all fields
- FROM
  - Section where tables are designated and joined
- WHERE (Optional)
  - Section for any filters or parameters

# Very Basic Accela Query

```
SELECT  B.B1_ALT_ID,
        B.SERV_PROV_CODE,
        B.B1_PER_ID1,
        B.B1_PER_ID2,
        B.B1_PER_ID3,
        B.B1_PER_GROUP,
        B.B1_PER_TYPE,
        B.B1_PER_SUB_TYPE,
        B.B1_PER_CATEGORY,
        B.B1_FILE_DD,
        B.B1_APPL_STATUS,
        B.B1_APPL_STATUS_DATE

FROM    B1PERMIT B

WHERE   B.SERV_PROV_CODE = '[Insert Client ID]'
AND     B.REC_STATUS = 'A'
```

This query gives a view of some of the frequently used fields in the B1PERMIT table

- B1_APPL_STATUS is often a filter in queries to omit statuses such as ‘VOID’
- B1_PER_GROUP / B1_PER_TYPE / B1_PER_SUB_TYPE / B1_PER_CATEGORY make up the Record Type and are also commonly used as a filter

# Basic SQL Syntax

- SELECT
  - Precede each field with table name or alias
    - **A.**FIELD1
    - **TABLE1.**FIELD1
  - Field aliases follow field names
    - A.FIELD1 **MyFieldName**
    - A.FIELD1 **as MyFieldName**
- FROM
  - Table aliases follow table names
    - TABLE1 **A**
    - TABLE1 **as A**
  - JOINS
    - INNER JOIN or JOIN: Forms a solid join between two tables. Only records with data in both tables will be returned.
    - LEFT OUTER JOIN or LEFT JOIN: Forms a join where data is not required in the table following the join, returning blanks for any applicable fields.
  - After the join list each join item starting with an “On” and following with an “And” for each additional join item
- WHERE (Optional)
  - Filters are separated with AND
  - Parameters are referenced with an “@” symbol

```
-- 1. SELECT: List of fields
Select  A.FIELD1,
        B.FIELD2,
        C.FIELD3
-- 2. FROM: Tables and table joins
From    TABLE1 A
Join    TABLE2 B
On      A.RECORD_ID = B.RECORD_ID
And     A.STATUS = B.STATUS
Left Join TABLE3 C
On      A.RECORD_ID = C.RECORD_ID
-- 3. WHERE: Filters (Optional)
Where   A.RECORD_ID = @PARAMETER_ID
And     (   A.STATUS = 'Active' Or
            A.STATUS = 'New')
```

# Basic Syntax with Accela

```
SELECT      B.B1_ALT_ID RECORD_ID,
            PD.TOTAL_FEE,
            PD.TOTAL_PAY,
            O.B1_OWNER_FULL_NAME

FROM        B1PERMIT B

-- Join: Most joins use these five fields
JOIN        BPERMIT_DETAIL PD
ON          PD.SERV_PROV_CODE=B.SERV_PROV_CODE
AND         PD.B1_PER_ID1=B.B1_PER_ID1
AND         PD.B1_PER_ID2=B.B1_PER_ID2
AND         PD.B1_PER_ID3=B.B1_PER_ID3
AND         PD.REC_STATUS=B.REC_STATUS

-- Left Join: Will return blanks if no owner listed
LEFT JOIN   B3OWNERS O
ON          O.SERV_PROV_CODE=B.SERV_PROV_CODE
AND         O.B1_PER_ID1=B.B1_PER_ID1
AND         O.B1_PER_ID2=B.B1_PER_ID2
AND         O.B1_PER_ID3=B.B1_PER_ID3
AND         O.REC_STATUS=B.REC_STATUS
AND         O.B1_PRIMARY_OWNER = 'Y'

-- Where: The first two should always be included
WHERE       B.SERV_PROV_CODE = '[Insert Client ID]'
And         B.REC_STATUS = 'A'
AND         B.B1_ALT_ID = @RecordID
```

**Comments:**

**Compound Key:** The fields SERV_PROV_CODE, B1_PER_ID1, B1_PER_ID2, and B1_PER_ID3 are ALWAYS used together in joins where the fields exist to maximize indexing.

**SERV_PROV_CODE:** This field needs to ALWAYS be set to the client id and used in all joins.

**REC_STATUS:** This field needs to be set to ‘A’ for Active since deleted records remain in the database with an ‘I’. It is ALWAYS used in joins for any table which has the field (or at least set to ‘A’ in any join)

# Summary Syntax

- Common summary functions are SUM, COUNT, MAX, MIN, etc.
- Fields not summarized need to appear in GROUP BY field
- ORDER BY appears if an order is desired (can be used in any query, not just summaries)

```
-- 1. SELECT: List of fields
Select  A.FIELD1,
        A.FIELD2,
        SUM(B.FIELD1) TOTAL_AMOUNT
-- 2. FROM: Tables and table joins
From    TABLE1 A
Join    TABLE2 B
On      A.RECORD_ID = B.RECORD_ID
-- 3. WHERE: Filters (Optional)
Where   A.RECORD_ID = @PARAMETER_ID
And     (   A.STATUS = 'Active' Or
            A.STATUS = 'New')
-- 4. GROUP BY: Non-summarized (Optional)
Group By A.FIELD1,
        A.FIELD2
-- 5. ORDER BY: Orders records (Optional)
Order By A.FIELD2
```

# Summary Syntax with Accela

```
-- 1. SELECT: All of the fields returned by the query
Select  B.B1_ALT_ID,
        SUM(FI.GF_FEE) AS TOTAL_FEES
-- 2. FROM: Tables and table joins
From    B1PERMIT B
Join    F4FEEITEM FI
On      B.SERV_PROV_CODE = FI.SERV_PROV_CODE
And     B.B1_PER_ID1 = FI.B1_PER_ID1
And     B.B1_PER_ID2 = FI.B1_PER_ID2
And     B.B1_PER_ID3 = FI.B1_PER_ID3
And     B.REC_STATUS = FI.REC_STATUS
And     FI.GF_ITEM_STATUS_FLAG IN ('NEW','INVOICED')
-- 3. WHERE: Filters and parameters (Optional)
Where   B.SERV_PROV_CODE = '[Insert Client ID]'
And     B.REC_STATUS = 'A'
And     B.B1_ALT_ID = @RecordID
-- 4. GROUP BY: Fields not summarized (Optional)
Group By B.B1_ALT_ID
-- 5. ORDER BY: Fields ordered by (Optional)
Order By B.B1_ALT_ID
```

# Accela Query Reminders

- Make the query as efficient and fast as possible
- Verify that indexes are used
- ALWAYS use the compound key otherwise your query may be slow
- If you are unsure about where to find a field search the Data Dictionary
- ALWAYS set the SERV_PROV_CODE field in your query to distinguish your data
- Set your REC_STATUS to A and include it in any joins where it exists

Filters:

- You may need to set your B1PERMIT.B1_APPL_STATUS to filter out statuses such as ‘VOID’
- Many tables have status columns to be aware of for filtering out data
- B1PERMIT stores the Group/Type/SubType/Category fields also used for filtering

# Using Accela Functions

```
SELECT  B.B1_ALT_ID RECORD_ID,
        --Parcel
        dbo.FN_GET_PARCEL_NBR(B.SERV_PROV_CODE,B.B1_PER_ID1,B.B1_PER_ID2,B.B1_PER_ID3) PARCEL_NUMBER,
        --Property Address
        dbo.FN_GET_PRI_ADDRESS_PARTIAL(B.SERV_PROV_CODE,B.B1_PER_ID1,B.B1_PER_ID2,B.B1_PER_ID3) PROPERTY_ADDRESS,
        --ASI Fields
        dbo.FN_GET_APP_SPEC_INFO(B.SERV_PROV_CODE,B.B1_PER_ID1,B.B1_PER_ID2,B.B1_PER_ID3,'Zone') ZONE

FROM    B1PERMIT B

WHERE       B.SERV_PROV_CODE = '[Insert Client ID]'
And         B.REC_STATUS = 'A'
AND         B.B1_ALT_ID = @RecordID
```

- Accela functions are used in your SELECT statement to return data more easily
- They always require an alias field name
- Most use the Compound Key
- Some have extra fields required that are outlined in the Function Reference Guide
- Be sure to put a “dbo.” before the function call

# Accela Function Reminder

Functions are used to extract a lot of information including:

- Contact Information
- Owner Information
- Parcel Number and Attributes
- ASI Fields
- Address Information

Commonly Used Functions

- FN_GET_APP_SPECIFIC_INFO: Used to pull ASI data
- FN_GET_PRI_ADDRESS_FULL: Returns full primary address in multi-line format
- FN_GET_PRI_ADDRESS_PARTIAL: Returns the first line of the address
- FN_GET_CONTACT_INFO: Returns various contact level information
- FN_GET_OWNER_INFO: Returns various owner level information
- FN_GET_PARCEL_NBR: Returns the parcel number

# Useful SQL Functions and Sites

**** I find it useful to search the web for SQL functions (Transact or T-SQL). Some sites I found that may be helpful are:**
**http://www.w3schools.com/sql/default.asp** ***This one has a nice DEMO area to try out queries as you learn***
**http://msdn.microsoft.com/en-us/library/bb510741**

| Function | Syntax | Use | Description |
|---|---|---|---|
| ISNULL() | ISNULL(Field to Check, Value) | ISNULL(A.AMOUNT,0) | Replaces a NULL field value with a default value |
| REPLACE() | REPLACE(Field to Scan, Value to Replace, Replacement) | REPLACE(A.FLAG,'Y','Yes') | Replaces any part of a field with another value |
| UPPER() OR LOWER() | UPPER(Field) | UPPER(A.NAME) | Makes a field either all uppercase or all lowercase |
| GETDATE() | GETDATE() | GETDATE() | Returns the current date |
| LENGTH() | LENGTH(Field) | LENGTH(A.FIELD) | Returns the length of a field |
| SUBSTRING() *look at LEFT() or RIGHT() too* | SUBSTRING(Field, Start Position, Length) | SUBSTRING(A.ADDRESS,1,3) | Returns a substring of a field |
| DATEADD() | DATEADD(Date Part, Number, Field) | SELECT DATEADD(month, -1, getdate()) | Adds or subtracts a set date part from a date (days, weeks, months, etc.) |
| LIKE | LIKE 'FIELD%' | WHERE A.ADDRESS LIKE 'PO%' | Allows you to use a wildcard '%' and search for partials |
| DISTINCT | SELECT DISTINCT Field | SELECT DISTINCT AT.B1_PER_GROUP FROM R3APPTYP AT | Allows you to select only distinct values |

# Accela Queries

- Exercise A
- Financial Reporting
- Example Fee Item Query
- Exercise B
- Using Sub-selects
- Using Sub-selects for Summaries
- Parameter Tips

# Exercise A

Time to try some things out. Create an example query with these Accela fields using the Data Dictionary and Function Reference Guide to help you:

1. Record/CAP ID (B1PERMIT table)
2. Property Address (Function)
3. Applicant Name (Function)
4. Current Status (B1PERMIT table)
5. Project Description (BWORKDES table)
6. Received/Open Date (B1PERMIT table)

# Exercise A – One Solution

```
SELECT      B.B1_ALT_ID as RECORD_ID,
            dbo.FN_GET_PRI_ADDRESS_PARTIAL(B.SERV_PROV_CODE,B.B1_PER_ID1,B.B1_PER_ID2,B.B1_PER_ID3)
                as PROPERTY_ADDRESS,
            dbo.FN_GET_CONTACT_INFO(B.SERV_PROV_CODE,B.B1_PER_ID1,B.B1_PER_ID2,B.B1_PER_ID3,
                'Applicant',NULL,'B','FullName',NULL,'U') as CONTACT_NAME,
            B.B1_APPL_STATUS as RECORD_STATUS,
            WD.B1_WORK_DESC as PROJECT_DESCRIPTION,
            B.B1_FILE_DD as OPEN_DATE

FROM        B1PERMIT B

JOIN        BWORKDES WD
ON          WD.B1_PER_ID1 = B.B1_PER_ID1
AND         WD.B1_PER_ID2 = B.B1_PER_ID2
AND         WD.B1_PER_ID3 = B.B1_PER_ID3
AND         WD.REC_STATUS = B.REC_STATUS
AND         WD.SERV_PROV_CODE = B.SERV_PROV_CODE

WHERE       B.REC_STATUS = 'A'
AND         B.SERV_PROV_CODE = '[Insert Client ID]'
AND         B.B1_APPL_STATUS NOT IN ('Void','Terminated')
```

# Financial Reporting

- Look at the ERD diagrams in the Data Dictionary to get an idea of how tables relate and what fields are used to join between them.
- Most will have multiple records per permit
- ACCOUNTING_AUDIT_TRAIL stores transaction level payment information (as does X4PAYMENT_FEEITEM)
  - Filter the ACTION field in your queries:
    - AAT.ACTION IN ('Payment Applied','Refund Applied','Void Payment Applied')
- F4FEEITEM stores all fee item information
  - Filter the status in your queries:
    - FI.GF_ITEM_STATUS_FLAG IN ('NEW','INVOICED')

# Example Fee Item Query

```
SELECT  B.B1_ALT_ID,
        -- Table Sequence IDs for verification
        FI.FEEITEM_SEQ_NBR,
        A.PAYMENT_SEQ_NBR,
        -- Fee Item Information
        FI.GF_ITEM_STATUS_FLAG,
        FI.GF_DES,
        FI.GF_FEE,
        --Payment/Refund Information
        A.TRAN_AMOUNT AMOUNT_PAID,
        FI.GF_FEE - ISNULL(A.TRAN_AMOUNT,0) AMOUNT_DUE,
        A.ACTION
FROM    B1PERMIT B
-- Get each fee item
JOIN    F4FEEITEM FI
ON      FI.SERV_PROV_CODE=B.SERV_PROV_CODE
AND     FI.B1_PER_ID1=B.B1_PER_ID1
AND     FI.B1_PER_ID2=B.B1_PER_ID2
AND     FI.B1_PER_ID3=B.B1_PER_ID3
AND     FI.REC_STATUS=B.REC_STATUS
AND     FI.GF_ITEM_STATUS_FLAG IN ('NEW','INVOICED')
-- Get any payments associated with a fee item
LEFT JOIN   ACCOUNTING_AUDIT_TRAIL A
ON      A.SERV_PROV_CODE=B.SERV_PROV_CODE
AND     A.B1_PER_ID1=B.B1_PER_ID1
AND     A.B1_PER_ID2=B.B1_PER_ID2
AND     A.B1_PER_ID3=B.B1_PER_ID3
AND     A.REC_STATUS=B.REC_STATUS
AND     A.FEEITEM_SEQ_NBR=FI.FEEITEM_SEQ_NBR
AND     A.ACTION IN ('Payment Applied','Refund Applied','Void Payment Applied')
WHERE   B.REC_STATUS = 'A'
AND     B.SERV_PROV_CODE = '[Insert Client ID]'
```

# Exercise B

Create a summary financial query for a report with these fields. Be sure to look at the ERD diagrams in the Data Dictionary:

1. Record/CAP ID (B1PERMIT table)
2. Record Status (B1PERMIT table)
3. Record Balance (BPERMIT_DETAIL table)
4. Receipt Number (F4PAYMENT – or F4RECEIPT for Customized Receipt Number)
5. Sum of Payments for the Receipt Number (ACCOUNT_AUDIT_TRAIL table – F4PAYMENT and others can be used as well)

# Exercise B – One Solution

```
SELECT  B.B1_ALT_ID RECORD_ID,
        B.B1_APPL_STATUS RECORD_STATUS,
        PD.BALANCE,
        P.RECEIPT_NBR,
        R.RECEIPT_CUSTOMIZED_NBR,
        SUM (AAT.TRAN_AMOUNT) TOTAL_PAYMENTS

FROM    B1PERMIT B

JOIN    BPERMIT_DETAIL PD
ON      B.SERV_PROV_CODE = PD.SERV_PROV_CODE
AND     B.B1_PER_ID1 = PD.B1_PER_ID1
AND     B.B1_PER_ID2 = PD.B1_PER_ID2
AND     B.B1_PER_ID3 = PD.B1_PER_ID3
AND     B.REC_STATUS = PD.REC_STATUS

-- Only joining to get receipt number
JOIN    F4PAYMENT P
ON      B.SERV_PROV_CODE = P.SERV_PROV_CODE
AND     B.B1_PER_ID1 = P.B1_PER_ID1
AND     B.B1_PER_ID2 = P.B1_PER_ID2
AND     B.B1_PER_ID3 = P.B1_PER_ID3
AND     B.REC_STATUS = P.REC_STATUS
JOIN    F4RECEIPT R
ON      P.SERV_PROV_CODE = R.SERV_PROV_CODE
AND     P.RECEIPT_NBR = R.RECEIPT_NBR
AND     P.REC_STATUS = R.REC_STATUS

JOIN    ACCOUNTING_AUDIT_TRAIL AAT
ON      P.SERV_PROV_CODE = AAT.SERV_PROV_CODE
AND     P.B1_PER_ID1 = AAT.B1_PER_ID1
AND     P.B1_PER_ID2 = AAT.B1_PER_ID2
AND     P.B1_PER_ID3 = AAT.B1_PER_ID3
AND     P.REC_STATUS = AAT.REC_STATUS
AND     P.PAYMENT_SEQ_NBR = AAT.PAYMENT_SEQ_NBR --NEED SINCE F4PAYMENT ABOVE
AND     AAT.ACTION IN ('Payment Applied','Refund Applied','Void Payment Applied')

WHERE   B.SERV_PROV_CODE = '[Insert Client ID]'
AND     B.REC_STATUS = 'A'
GROUP BY B.B1_ALT_ID,
        B.B1_APPL_STATUS,
        PD.BALANCE,
        P.RECEIPT_NBR,
        R.RECEIPT_CUSTOMIZED_NBR
```

# Parameter Tips

- When building a report with a record id parameter (B1_ALT_ID), it may be easiest to hard code it in your query to a useful record, waiting until the query is fully built to point it to the parameter field.
  - Temporary: WHERE B.B1_ALT_ID = ‘ABC000123’
  - Final: WHERE B.B1_ALT_ID = @RecordID
- When using a date include a “<“ @Date + 1 to capture all time stamps for the given date.
  - WHERE B.B1_FILE_DD >= @ParameterDate AND B.B1_FILE_DD < @ParameterDate + 1
- For queries meant to populate a parameter list within a report, ALWAYS use a reference table instead of doing a select distinct from a transaction table.
  - RIGHT: SELECT DISTINCT R1_PER_GROUP FROM **R3APPTYP**
  - WRONG: SELECT DISTINCT B1_PER_GROUP FROM **B1PERMIT**

# Case Statements

*Case Statements are basically an extended if/then statement, allowing you to assign various values to a field. They are very useful for setting flags to be summed.*

General Syntax:

CASE WHEN (Case to evaluate) THEN (Value) ELSE (Optional) END

```
SELECT  P.SD_APP_DES,
        CASE    WHEN P.SD_APP_DES = 'Complete' THEN 1
                WHEN P.SD_APP_DES = 'Complete - No Plan Review' THEN 1
                ELSE 0 END FLAG_COMPLETE
FROM    GPROCESS P
WHERE   P.SD_PRO_DES = 'Application Submittal'
```

Alternate Syntax:

CASE (Item to evaluate) WHEN (Value to evaluate) THEN (Value) ELSE (Optional) END

```
SELECT  P.SD_APP_DES,
        CASE    P.SD_APP_DES
                WHEN 'Complete' THEN 1
                WHEN 'Complete - No Plan Review' THEN 1
                ELSE 0 END FLAG_COMPLETE
FROM    GPROCESS P
WHERE   P.SD_PRO_DES = 'Application Submittal'
```

# Using Sub-selects

- If you are joining to a table and only returning **one field**, a sub-select may be more efficient.

*They will not work if multiple records are found.*

Before: BWORKDES is accessed with a join

```
SELECT  B.B1_ALT_ID RECORD_ID,
        --Work Description
        WD.B1_WORK_DESC

FROM    B1PERMIT B
JOIN    BWORKDES WD
ON      WD.B1_PER_ID1 = B.B1_PER_ID1
AND     WD.B1_PER_ID2 = B.B1_PER_ID2
AND     WD.B1_PER_ID3 = B.B1_PER_ID3
AND     WD.SERV_PROV_CODE = B.SERV_PROV_CODE
AND     WD.REC_STATUS = B.REC_STATUS
WHERE   B.SERV_PROV_CODE = '[Insert Client ID]'
And     B.REC_STATUS = 'A'
AND     B.B1_ALT_ID = @RecordID
```

After: BWORKDES is accessed with a sub-select

```
SELECT  B.B1_ALT_ID RECORD_ID,
        --Work Description
        (   Select  WD.B1_WORK_DESC
            From    BWORKDES WD
            Where   WD.B1_PER_ID1 = B.B1_PER_ID1
            And     WD.B1_PER_ID2 = B.B1_PER_ID2
            And     WD.B1_PER_ID3 = B.B1_PER_ID3
            And     WD.SERV_PROV_CODE = B.SERV_PROV_CODE
            And     WD.REC_STATUS = B.REC_STATUS) WORK_DESCRIPTION

FROM    B1PERMIT B
WHERE   B.SERV_PROV_CODE = '[Insert Client ID]'
And     B.REC_STATUS = 'A'
AND     B.B1_ALT_ID = @RecordID
```

# Using Sub-selects for Summaries

- Sub-selects are also useful if you are summarizing one field. Instead of grouping every field in the query to get one summary field, use a Sub-select

Before: F4FEEITEM is joined in the main query

```
-- 1. SELECT: All of the fields returned by the query
Select  B.B1_ALT_ID,
        SUM(FI.GF_FEE) AS TOTAL_FEES
-- 2. FROM: Tables and table joins
From    B1PERMIT B
Join    F4FEEITEM FI
On      B.SERV_PROV_CODE = FI.SERV_PROV_CODE
And     B.B1_PER_ID1 = FI.B1_PER_ID1
And     B.B1_PER_ID2 = FI.B1_PER_ID2
And     B.B1_PER_ID3 = FI.B1_PER_ID3
And     B.REC_STATUS = FI.REC_STATUS
And     FI.GF_ITEM_STATUS_FLAG IN ('NEW','INVOICED')
-- 3. WHERE: Filters and parameters (Optional)
Where   B.SERV_PROV_CODE = '[Insert Client ID]'
And     B.REC_STATUS = 'A'
And     B.B1_ALT_ID = @RecordID
-- 4. GROUP BY: Fields not summarized (Optional)
Group By B.B1_ALT_ID
```

After: F4FEEITEM is joined in a sub-query

```
SELECT  B.B1_ALT_ID,
        (   Select  SUM(FI.GF_FEE)
            From    F4FEEITEM  FI
            Where   FI.SERV_PROV_CODE = B.SERV_PROV_CODE
            And     FI.B1_PER_ID1 = B.B1_PER_ID1
            And     FI.B1_PER_ID2 = B.B1_PER_ID2
            And     FI.B1_PER_ID3 = B.B1_PER_ID3
            And     FI.REC_STATUS = B.REC_STATUS
            And     FI.GF_ITEM_STATUS_FLAG IN ('NEW','INVOICED')
            ) TOTAL_FEES
FROM    B1PERMIT B
WHERE   B.SERV_PROV_CODE = '[Insert Client ID]'
AND     B.REC_STATUS = 'A'
AND     B.B1_ALT_ID = @RecordID
```