Oracle BRM Training

By Nedhya Sharma

AGENDA

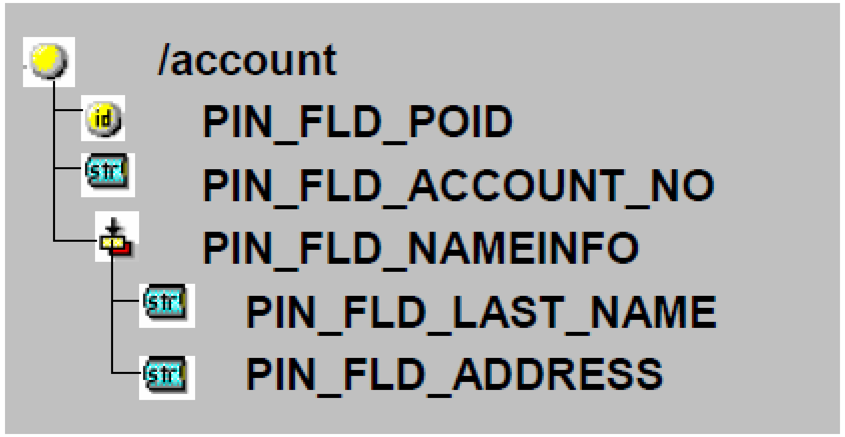
1. Storable Classes & Objects
2. BRM Simple Datatypes
3. BRM Complex Datatypes

Storable class

A **Storable Class** defines the structure for a set of data to be stored in the BRM Database.
It is just like a java class, as the structure is created in database, it is named as storable class.
Storable class fields can span more than one table.
Examples of some of important storable classes.

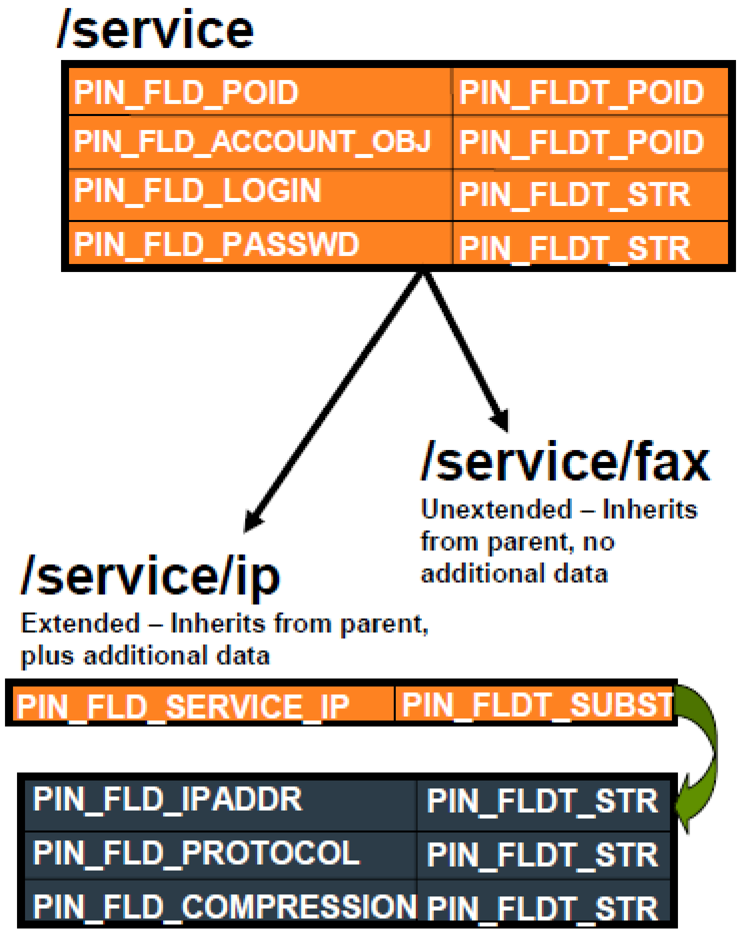
Storable class	Description	Fields span in tables
/account		
/balance_group		
/billinfo		

All storable class names start with "/".
Field names of storable classes are prefixed with PIN_FLD_XXXX. XXX stand for field name.



```
/account
    PIN_FLD_POID
    PIN_FLD_ACCOUNT_NO
    PIN_FLD_NAMEINFO
        PIN_FLD_LAST_NAME
        PIN_FLD_ADDRESS
```

Storable class naming convention



/service

PIN_FLD_POID	PIN_FLDT_POID
PIN_FLD_ACCOUNT_OBJ	PIN_FLDT_POID
PIN_FLD_LOGIN	PIN_FLDT_STR
PIN_FLD_PASSWD	PIN_FLDT_STR

/service/fax

Unextended – Inherits from parent, no additional data

/service/ip

Extended – Inherits from parent, plus additional data

PIN_FLD_SERVICE_IP	PIN_FLDT_SUBST

PIN_FLD_IPADDR	PIN_FLDT_STR
PIN_FLD_PROTOCOL	PIN_FLDT_STR
PIN_FLD_COMPRESSION	PIN_FLDT_STR

All storable class names start with forward slash “/”.

– Example: /service

Field names of storable classes are prefixed with PIN_FLD_XXXX. XXX stand for field name.

Subclasses are separated by another slash (/)

– Example: /service/ip, /service/fax

Subclass inherits parent class definition

– Unextended subclasses have no additional data; same class definition as parent class.

– Extended subclasses have extended data in addition to data defined by parent class.

Important Storable classes and it Uses

Storable Class	It's use
/account	Stores information about the customer, including contact names, address, status, and *customer segment* information. An **/account** object is linked to the following objects: Balance group objects that contain the account balances. Bill unit objects (**/billinfo**) that contain account billing information. Bill objects. A service object for each service that the account owns. An account can own any number of services. Additional account information stored in **/profile** objects.
/balance_group	Stores the balance information for various resources in an account such as dollars, free minutes, bytes, and frequent flyer miles. A balance group includes one or more sub-balances for each resource. The sub-balance contains the current amount, resource type, validity dates for the resource, rollover data, and sub-balance contributors.
/billinfo	Stores all billing, payment method, accounting cycle, payment collection date, and hierarchy information necessary to bill an account. A **/billinfo** object is created for every account.
/bill	Stores billing information, such as the amount due, amount adjusted, currency, and bill number.

Important Storable classes and it Uses

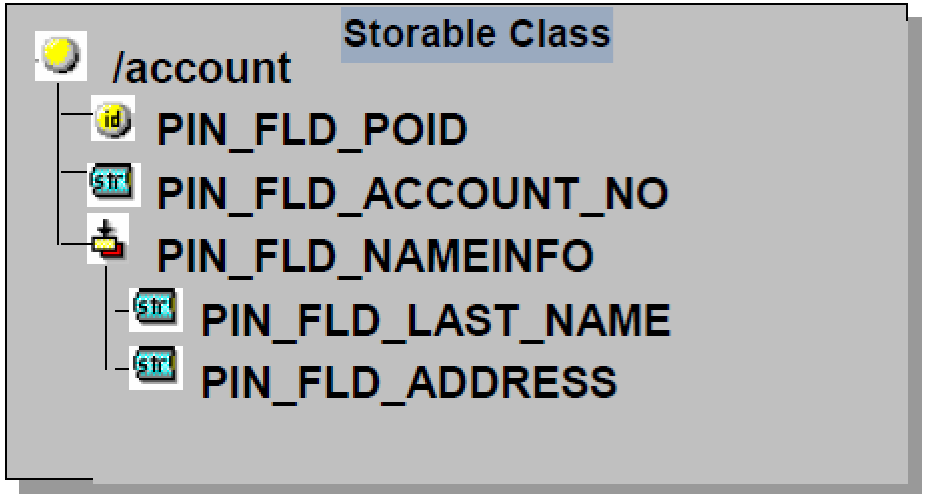
Storable Class	It's use
/item	Abstract class for storing *accounts receivable (A/R)* information. Subclasses of the **/item** object store different types of A/R information; for example, payments, adjustments, and cycle charges. Any impact to an account's A/R is stored in an **/item** object.
/invoice	Stores a customer invoice and information about the invoice, such as the bill it is associated with. Each **/bill** object can have a corresponding **/invoice** object.
/service	Abstract class to support subclasses for specific services, such as telephony or IP access. Subclasses define the properties that are specific to each service; for example, the telephony bearer service or the IP address.
/purchased_product	Stores information about purchased products for /account objects. Products owned by /account or any services in /account are stored in one or more of these objects. This object contains the reference to **/account**. **/account** has no references to this object.
/purchased_discount	Stores information about purchased discounts for /account objects. Discounts owned by an account or any services in the account are stored in one or more of these objects. This object stores the reference to the **/account** object; however, the **/account** object does not store a reference to this object.

Important pricing Storable classes and it Uses

Storable Class	It's uses
/product	Stores the information for a single product. Changes to a **/product** object affect all customers who own the product.
/deal	The **/deal** object includes information about each product in the deal, including valid dates and discounts for each product.
/plan	Stores information about a plan in a price list, such as the services and deals included in the plan. Also includes credit limit information.
/rate	Stores information about a single rate. Each **/rate** storable object is owned by a specific **/product** storable object and links directly to that product.

Storable Objects

A Storable object is an instance of the class, that is it represents actual set of data.
Example:
Student is a class and data with respective one student is an object.



Storable Class

```
/account
  PIN_FLD_POID
  PIN_FLD_ACCOUNT_NO
  PIN_FLD_NAMEINFO
    PIN_FLD_LAST_NAME
    PIN_FLD_ADDRESS
```

Storable Object

```
0.0.0.1 /account 12345 1
0.0.0.1-12345
NAMEINFO[1]
   Smith
   123 Elm Street
```

BRM Database Storage Model

Data modeling (data modelling) is the process of creating a data model for the data to be stored in a database. This data model is a conceptual representation of Data objects, the associations between different data objects, and the rules. BRM database is no exceptional.

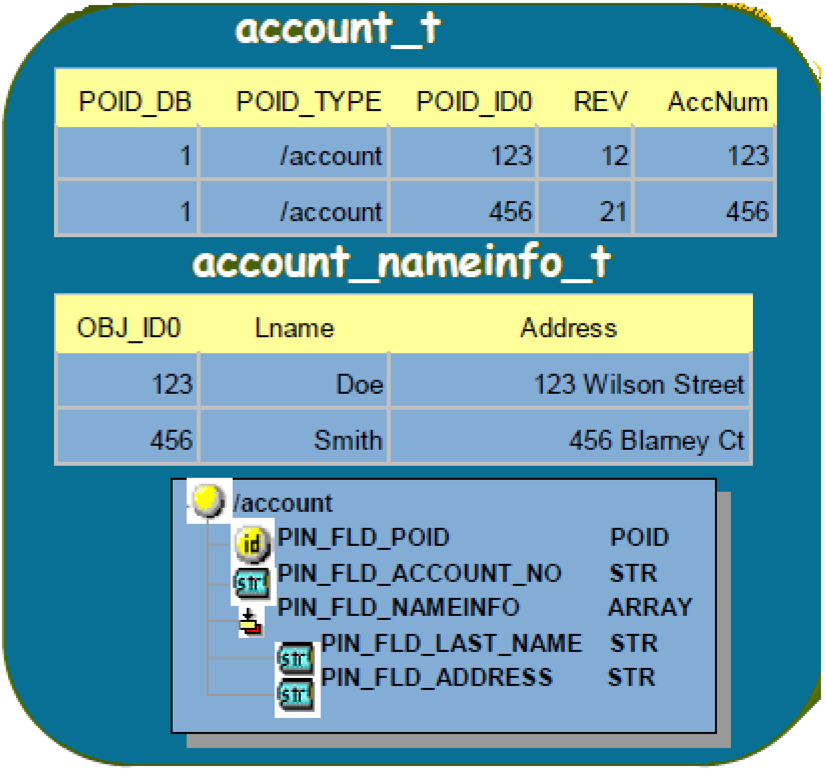
BRM Storage model consist of relational database and data dictionary

BRM Database

- Relational Database
- Tables and columns

Data Dictionary

- Storable Class Definitions
- Field Definitions



account_t

POID_DB	POID_TYPE	POID_ID0	REV	AccNum
1	/account	123	12	123
1	/account	456	21	456

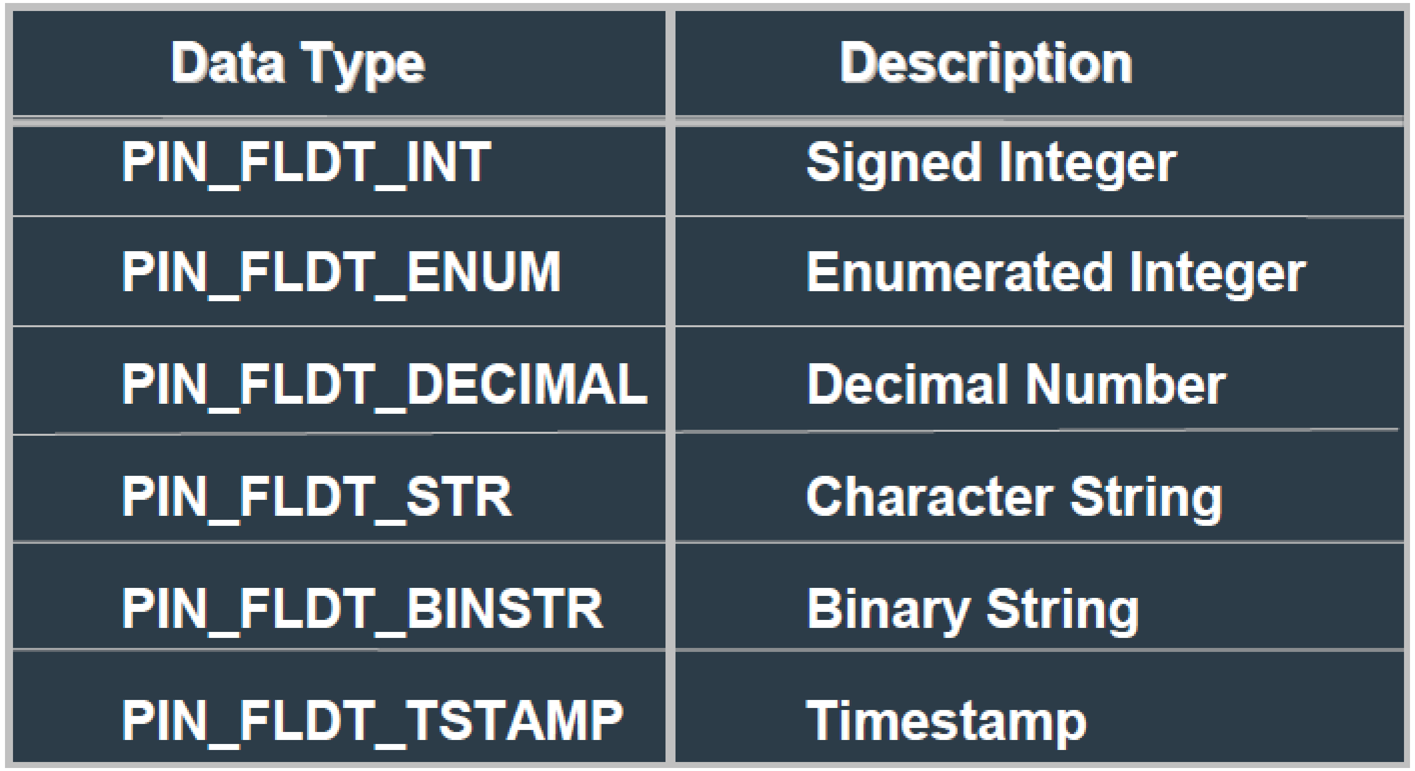
account_nameinfo_t

OBJ_ID0	Lname	Address
123	Doe	123 Wilson Street
456	Smith	456 Blamey Ct

```
/account
  PIN_FLD_POID              POID
  PIN_FLD_ACCOUNT_NO        STR
  PIN_FLD_NAMEINFO          ARRAY
     PIN_FLD_LAST_NAME      STR
     PIN_FLD_ADDRESS        STR
```

BRM Data Type

BRM has set of data types.
Simple data types generally correspond to C data types.



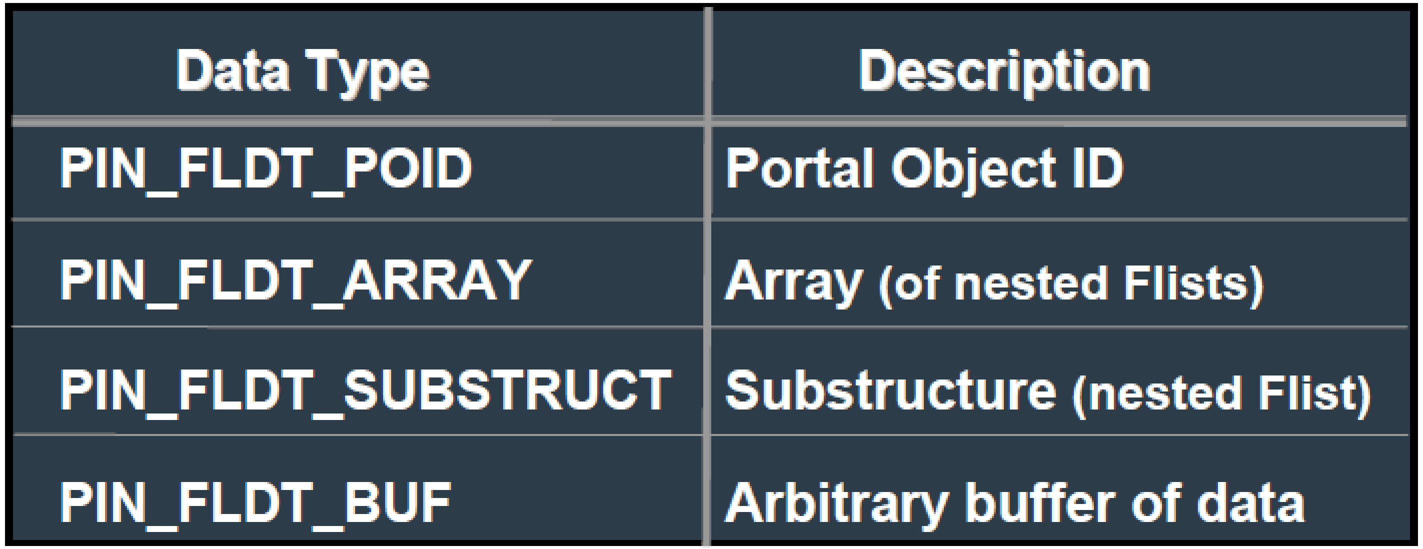
Data Type	Description
PIN_FLDT_INT	Signed Integer
PIN_FLDT_ENUM	Enumerated Integer
PIN_FLDT_DECIMAL	Decimal Number
PIN_FLDT_STR	Character String
PIN_FLDT_BINSTR	Binary String
PIN_FLDT_TSTAMP	Timestamp

Note: Fields should be defined in pcm_flds.h header file and also in data dictionary before they are used in flist.

BRM Data Type

The complex data types are specific to BRM

- POID (Portal Object ID)
- Array
- Substruct
- Buffer



Data Type	Description
PIN_FLDT_POID	Portal Object ID
PIN_FLDT_ARRAY	Array (of nested Flists)
PIN_FLDT_SUBSTRUCT	Substructure (nested Flist)
PIN_FLDT_BUF	Arbitrary buffer of data

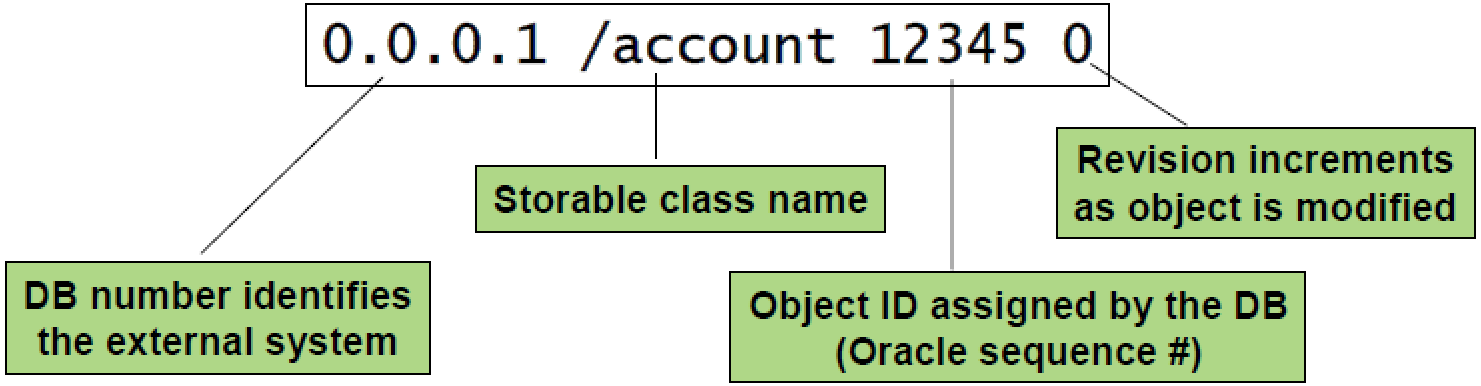
Note: Fields should be defined in pcm_flds.h header file and also in data dictionary before they are used in flist.

POID [Portal Object IDentifier]

POID – Represents a unique value. It is similar to primary key in database.
Uniquely identifies a set of data (storable object) in the BRM database.

A POID value contains 4 pieces of information

- Database number
- Object type (storable class name)
- Object ID
- Revision level



```
0.0.0.1 /account 12345 0
```

DB number identifies the external system

Storable class name

Object ID assigned by the DB (Oracle sequence #)

Revision increments as object is modified

BRM Flist

BRM application has developed using C, C++ and Java.

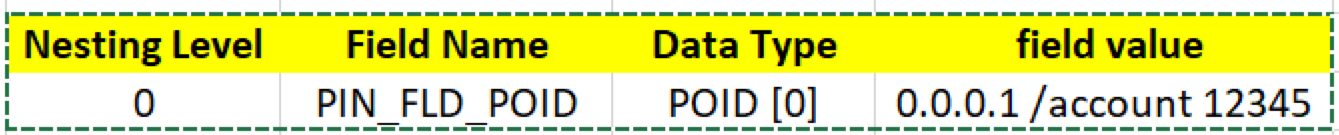
Simple data types generally correspond to C data types.

BRM business tier has a structure framework called Flist, which is used to process and pass data among different function modules internally.

Flist framework has list of fields and field values on it.

Flist framework has specifications which needs to be follow strictly to define the Flist.

Syntax of Flist Specification



Nesting Level	Field Name	Data Type	field value
0	PIN_FLD_POID	POID [0]	0.0.0.1 /account 12345

Nesting Level: Used to specify the field depth in the storable class hierarchy. All parent class fields will be at level 0 and immediate subclass or child class will be at the next level.

Field Name: Field name is the field's of storable class.

Data type: Any of the BRM data types

Field Value: Value of the field

Complex Data Type

SUBSTRUCT Data Type

A SUBSTRUCT is a reference to a nested Flist

Typically used with storable subclasses to define the extended set of data

– For performance reasons nested Flists should be limited to 3 levels

Used with Input and Output Flist specs to segregate sets of data

– No limit to nesting level

Array Data Type

An ARRAY References an array of nested Flists

Each Flist in the array is referenced by an Element ID

– Sparse arrays; element ID does not have to start with 0 or 1

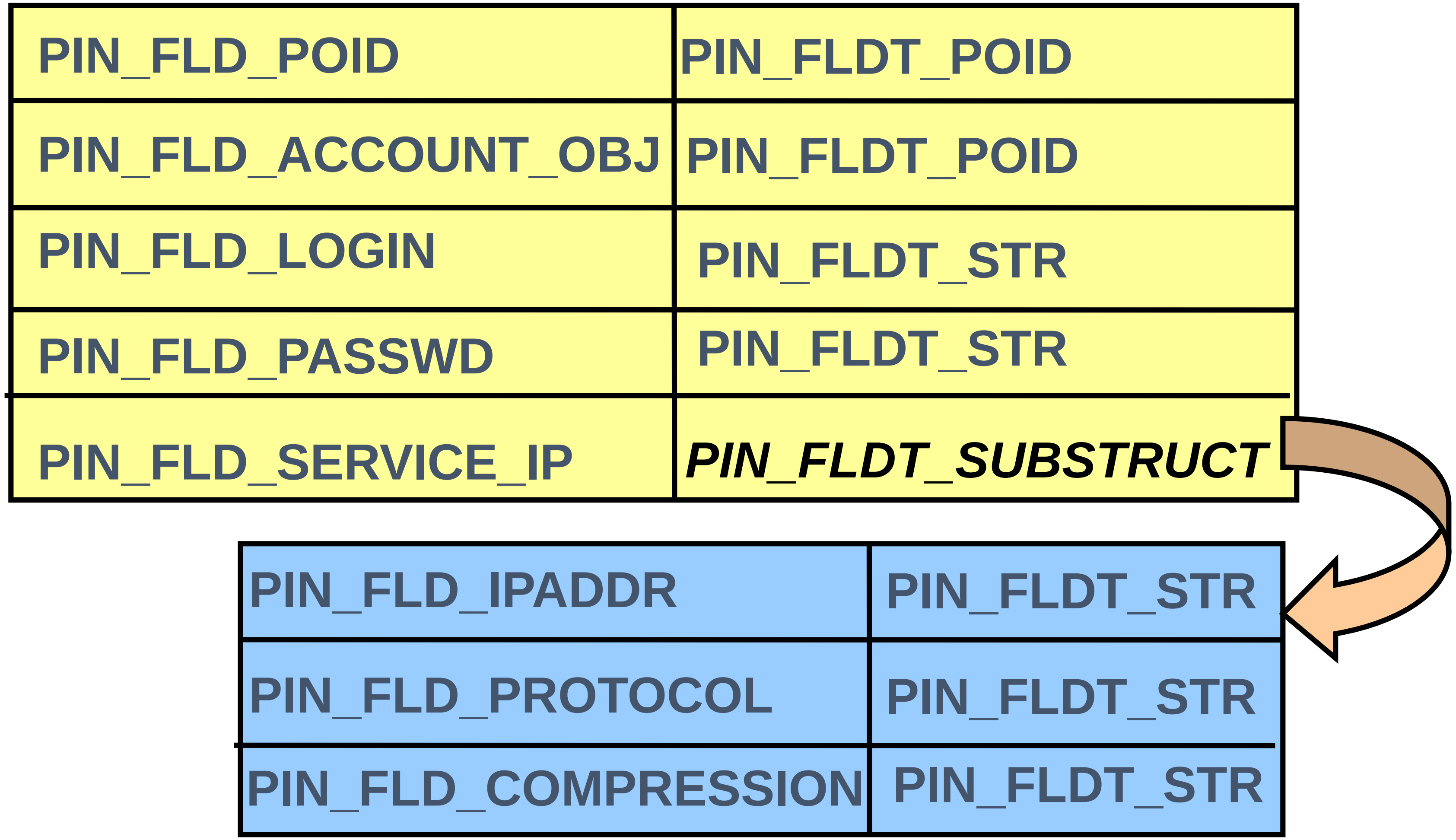
– Element IDs do not have to be sequential

Each element in the array shares the same Flist specification

– Field name, value, nesting level, permissions

– An array with 1 element is functionally equivalent to a substruct (refers to a single nested Flist)

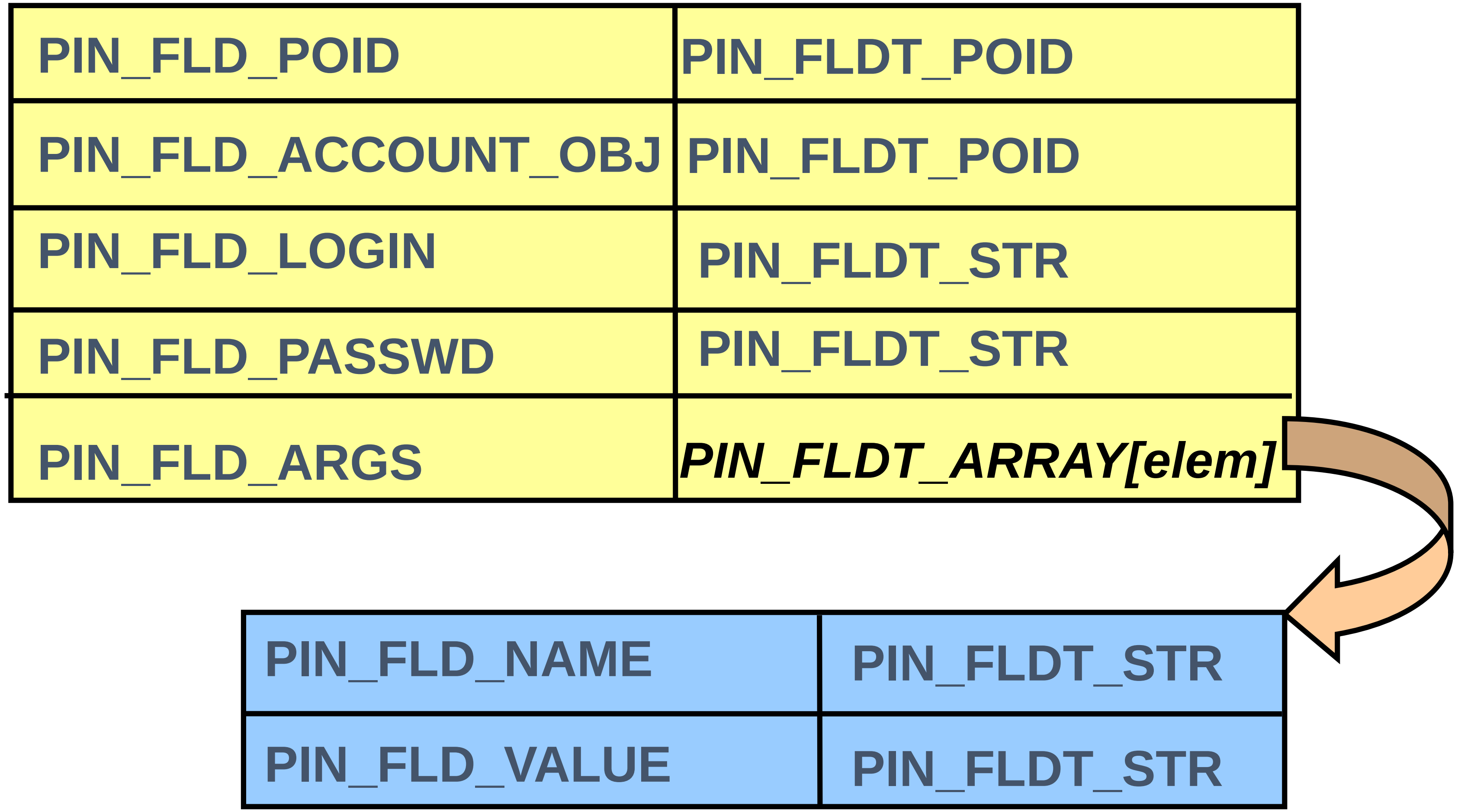
Complex Data Type – Substruct Example



PIN_FLD_POID	PIN_FLDT_POID
PIN_FLD_ACCOUNT_OBJ	PIN_FLDT_POID
PIN_FLD_LOGIN	PIN_FLDT_STR
PIN_FLD_PASSWD	PIN_FLDT_STR
PIN_FLD_SERVICE_IP	*PIN_FLDT_SUBSTRUCT*

PIN_FLD_IPADDR	PIN_FLDT_STR
PIN_FLD_PROTOCOL	PIN_FLDT_STR
PIN_FLD_COMPRESSION	PIN_FLDT_STR

Complex Data Type – Array Example



PIN_FLD_POID	PIN_FLDT_POID
PIN_FLD_ACCOUNT_OBJ	PIN_FLDT_POID
PIN_FLD_LOGIN	PIN_FLDT_STR
PIN_FLD_PASSWD	PIN_FLDT_STR
PIN_FLD_ARGS	*PIN_FLDT_ARRAY[elem]*

PIN_FLD_NAME	PIN_FLDT_STR
PIN_FLD_VALUE	PIN_FLDT_STR

Flist to Database Mapping Rules

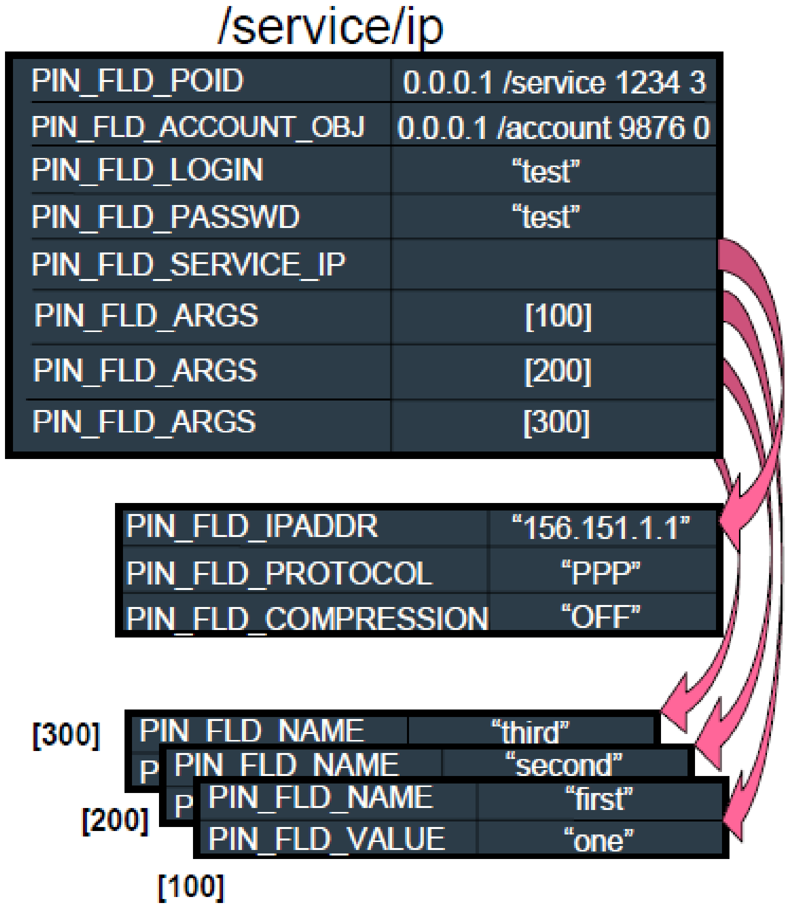
DM translates each Flist of data to database tables

– Base Flists, Substructs, Arrays, Buffers

Most Fields translate to one column name

– Exceptions:

- POIDs translate to 4 columns
- Substruct table has extra column for object ID from POID
- Array table has 2 extra columns for object ID from POID and element ID



/service/ip

PIN_FLD_POID	0.0.0.1 /service 1234 3
PIN_FLD_ACCOUNT_OBJ	0.0.0.1 /account 9876 0
PIN_FLD_LOGIN	“test”
PIN_FLD_PASSWD	“test”
PIN_FLD_SERVICE_IP	
PIN_FLD_ARGS	[100]
PIN_FLD_ARGS	[200]
PIN_FLD_ARGS	[300]

PIN_FLD_IPADDR	“156.151.1.1”
PIN_FLD_PROTOCOL	“PPP”
PIN_FLD_COMPRESSION	“OFF”

[300]

PIN_FLD_NAME	“third”

[200]

PIN_FLD_NAME	“second”

[100]

PIN_FLD_NAME	“first”
PIN_FLD_VALUE	“one”

service_t

poid_db	poid_type	poid_id0	poid_rev	login	passwd
0.0.0.1	/service/ip	1234	3	“test”	“test”
0.0.0.1	/service/ip	5678	6	“joe”	“joe”

service_ip_t

obj_id0	ipaddr	protocol	compression
1234	“156.151.1.1”	“PPP”	“OFF”
5678	“156.151.1.2”	“SLIP”	“ON”

service_ip_args_t

obj_id0	rec_id	name	value
1234	100	“first”	“one”
1234	200	“second”	“two”
1234	300	“third”	“three”
5678	100	“pop”	“timbuktu”
5678	200	“speed”	“28.8K”
5678	300	“drop line credit”	“no”

Exercise 3

1. What is a storable class?
2. What are the naming conventions of storable class?
3. What is Unextended storable class?
4. What is the use of extended storable class?
5. What is POID?
6. What is the use of ARRAY & SUBSTRUCT data types?
7. What is Flist?
8. Why do we use level in Flist specifications?
9. What is the difference between ARRAY & SUBSTRUCT?
10. POID is made up of how many values?

“Thank You”