시험문제: 80 문제

시험시간: 3 시간

Weighting per Topic (not listed for all exams):

+ = <8%

++ = 8-12%

+++ = >12%

Competency	Importance	Topic	Way(s) to attain		
			Primary	Alternative	Other
Articulate, explain, describe, and outline solutions and processes in...	60%	NetWeaver Overview +	TAW10	NW001	SAPTEC
		ABAP Workbench usage +	TAW10	BC400, BC401	
		ABAP debugger program usage +	TAW10	BC400, BC402	
		ABAP types and data objects ++	TAW10	BC400	
		Internal table definition and use +	TAW10	BC400	
		SQL statements incl. update strategies ++	TAW10, TAW11	BC400, BC414	
		ABAP Dictionary ++	TAW10	BC430	
		Unicode +	TAW11	BC402	
		Classical screens (dynpros) and selection screens +	TAW10	BC400	
		ALV Grid Control +	TAW11E	BC405	
		User Interfaces (Web Dynpro) +	TAW12	NET310	
Build, implement, configure, model, and troubleshoot solutions and processes in...	40%	**Basic ABAP programs and interfaces creation +++**	TAW10	BC400, BC402	
		Class identification analysis and design ++	TAW12	BC401	
		Enhancement and Modifications ++	TAW12	BC425, BC427	
		Table relationships +	TAW10	BC430	

Material Agenda.

1 주차 [TAW10_1-1, TAW10_1-2]	1. The Big Picture
	2. Navigation / The System Kernel
	3. Communication&Integration Technologies
	4. Flow of an ABAP Program / Introduction to the ABAP / Basic Elements
	5. Modularization
	6. Complex Data Object
	7. Data Modeling&Data Retrieval / ABAP Open SQL
	8. Selecetion Screen / Programs Calls&Data Storage Manage / New TEST Tools
2 주차 [TAW10_2-1, TAW10_2-2]	1. Introduction to Screen Programming
	2. The Program Interface
	3. Screen Elements for Input/Output
	4. Screen Elements: Subscreens and Tabstrip Controls
	5. Introduction to the Dictionary / Data objects in the ABAP Dictionary
	6. Performance When Accessing Tables
	7. Input Checks / Object Dependencies / Changing Tables
	8. Views and Maintenance Diglogs
	9. Search Help
3 주차 [BC405, BC414]	1. Introduction / Selection Screen
	2. Logical Databases / Data Retrieval Without Logical Databases
	3. Call Programs with Data Transfer
	4. **Appendix: ALV Grid Control**
	5. Database Updates with Open SQL / LUWs&Client/Server Architecture
	6. SAP Locking Concept
	7. Organizing Database Updates
	8. Complex LUW Processing
	9. Appendix
4 주차 [TAW12_1]	1. Introduction to Object-Oriented Programming
	2. Object-Oriented Concepts & Programming Techniques
	3. Object-Oriented Repository Objects
	4. Class-Based Exception Concept
	5. Shared Objects
	6. Dynamic Programming
5 주차 [TAW12_2]	1. Changing the SAP Standard System / Enhancing Dictionary Elements
	2. Enhancements Using Customer Exits
	3. Business Add-Ins
	4. Modifications
	5. **Enhancements** / Implicit Enh / BADI
	6. Web Dynpro: Introduction
	7. WebDynpro Controllers / Context at Design Time /Defining the UI
	8. Controller and Context Programming
	9. Internationalization and Messages

SAP – " Systems, Applications and Products in data processing"

ABAP – Advanced Business Applications Programming

자주 쓰는 단축키: F3 (Back), F8 (프로그램 실행)

<1주차 [TAW10_1-1, TAW10_1-2]>

1. The Big Picture

- SAP의 Products: SAP Business One(=B1) < SAP Business ByDesign < SAP Business All-in-One(=A1) < SAP Business Suite
- SAP Business suite의 구조: Composite applications/SAP applications/SAP NetWeaver로 구성.
  - SAP applications: 단위 프로그램들이 뭉쳐져 있는 것. 모듈을 위한 용도에 맞는 프로그램들의 집합
  - SAP NetWeaver: 상품화된 SAP의 제품 이름.
- **Component**: Application들을 제품화 시켜놓은 것. 각 모듈에서 필요한 기능을 컴포넌트에서 가져다 씀. 중복된 모듈에서 사용 가능.
- SAP ECC(Enterprise Central Component): Central하게 필요한 applications를(default로 필요한) 기능관련성에 관계 없이 묶어놓은 것.
- SAP 버전: 7.100이 최신.

< SAP Netweaver >

: **모듈을 NetWeaver가 감싸고 있고 Netwaver가 non-SAP와 통신이 가능하게 해 줌.**

- NetWeaver의 integration layers: People integration(사용자 인터페이스 관련), Information Integration(회사에 대한 정보들), Process Integration(업무처리 프로세스 등), Application Paltform(NetWeaver Application Server는 여기에 존재)
- SAP NetWeaver Application Server
  - SAP ECC, SAP NW Portal, SAP CRM, SAP SRM 등은 모두 SAP NetWeaver AS(application server)를 default로 써야한다.
  - NetWeaver WAS(Web application server)는 3가지로 구성
    - Presentation layer – ex) SAP GUI, Browser
    - Application layer – JAVA/ABAP/JAVA&ABAP
    - Database layer – 여러 DB system 지원함

2. Navigation

- SAP GUI 종류 : 터미널로 SAP 시스템에 접근. (Client 환경에서)
  - SAP GUI for the Windows
  - SAP GUI for JAVA
  - SAP GUI for HTML
- Server 환경
  - Dispatcher: 각 Client에서의 요청에 대한 교통흐름 제어
  - ICM(Internet Communication Manager): SAP환경이 Web에서도 동작하도록 도와주는 하나의 Demon. SAP GUI for HTML과 Dispatcher 연결해줌.
- SAP GUI 로그온 시 첫 화면: SAP Easy Access화면.
- SAP user interface (교재 p.43) : 뭔지는 봐두자.
  - Command field
  - Menu bar
  - System toolbar : 변경 불가
  - Title bar
  - Applications toolbar: 사용자가 변경 가능
  - Tab pages
  - Status bar
- SAP GUI 로그오프하기.
  - 메뉴바에 노란 화살표 아이콘 누르기
  - '/nend'또는 '/nex' 입력. ('/nex'는 물어보지 않고 바로 종료)
  - 메뉴에서 System->Log off 선택
- Easy Access 화면의 왼쪽
  - Favorites와 Role-based menu(해당 사용자의 권한에 따라 메뉴 체계가 틀리다.)있음.
- Function들 호출하는 방법
  - **Ok code field**에 T-code입력
  - 메뉴 바에서의 item 선택
  - 즐겨찾기 리스트에서 더블클릭
- System -> status: 내가 현재 접속하고 있는 system에 대한 개략적인 정보가 나옴.
- Navigation keys
  - TAB: field group에서 field element간 이동 시
  - Ctrl+TAB: 다음 field group의 첫번째 field element로 이동
  - CTRL+/: Ok code field로 이동
- Command
  - /n:현재 transaction을 취소
  - /nXXXX(XXXX는 t-code): 현재 실행 중이던 session은 종료하고 XXXX를 실행
  - /o: 현재 session 살려두고 새로 session 창 열 때
  - /oXXXX: 현재 session 살려두고 새로 XXXX를 수행할 session을 열 때
  - /nend: 현재 session을 종료할 때. Confirmation dialog box가 뜸
  - /nex: 현재 session을 confirm없이 종료할 때
  - /i: 현재 사용 중인 session만 종료할 때
- Help
  - F1: 해당 input field의 의미 정보 알려줌
  - F4: 해당 input field의 possible entries(후보 입력 값)에 대한 도움을 줌.
- SAP Library(Online documentation) : system function, release 내용 등에 대한 정보 얻고 싶을 때
  - http://help.sap.com
  - http://sdn.sap.com

3. System Kernel

- Single-tier/two-tier/three-tier configuration
  - Single-tier: 모든 프로세스 task(DB, application, presentation process)가 하나의 컴퓨터에서 수행된다.
  - Two-tier: App/DB가 한 pc에, presentation은 특별한 다른 서버에서 수행. 직렬로 연결됨. App/DB 서버에 접근이 로드가 걸리면 접근이 막힐 수 있으므로 추천하지 않음.
  - Three-tier: 각각의 layer가 자신의 host를 가짐. (DB는 논리적으로 하나). 가장 많이 사용. 서버가 부족할 때 application layer의 server를 쉽게 늘릴 수 있다.
- 대부분의 Client/Server configurations (**기출!**) : 3개의 tier가 있어서 'SAP R/3'라고 부름.
  - Presentation processes: 사용자가 직접 접하는 환경 (ex. 개인 pc)
  - Application processes: SAP GUI를 통해 Client가 서비스를 요청하면 이를 처리하고 서비스를 제공해주는 layer

- Database Processes: 사용자가 요청할 data가 들어 있다.

- The instance
  - (SAP) instance(논리적 개념)와 Application Server(H/W적 개념)은 유사한 의미.
  - 한 개 이상의 서비스를 제공하는 SAP System component를 묶은 관리상의 unit.
- SAP System의 구조 (외우기!)
  - Presentation layer(level): SAP GUI 또는 Web browser
  - Application level: load balancing을 위해 여러 컴퓨터 사용.
    - **Central instance**
      - AS ABAP에서 **Message Server와 Enqueue work processor를 갖고 있는 instance가 central instance임.**
      - AS JAVA에서 Software Deployment Manager를 갖고 있는 instance가 central instance임.
      - Enqueue instance: DB에 job 수행 시, Lock을 잡고 해제하는 역할을 수행.
      - 만약 central instance와 DB가 같은 pc에 깔리면, 그건 **central system**이라고 한다.
      - 나머지는 대개 dialog instance임.
    - Central service instance
      - AS JAVA에서 Message Service와 Enqueue service를 가진 녀석이 Central service instance.
      - SDM을 가진 것이 Central instance.
  - Database level

- AS ABAP Processes
  - 가장 상위: communication의 역할. (*아래는 work process들이 아님!)*
    - ABAP **message server(MS)**: user가 많을 경우 필요. 제일 놀고 있는 서버에 연결시켜줌. Dispatcher들과 통신을 해서 **load 밸런싱**을 수행함.
    - Gateway(GW) : SAP system이 외부 AS와 통신할 수 있도록 해주는 server
    - **ICM**(Internet Communication Manager): SAP system 안에 존재. Web환경을 지원하기 위한 하나의 매니저. Demon. SAP 시스템이 web 프로토콜(HTTP같은)을 사용할 수 있도록 해줌.
  - 중간 (중간/하위는 SAP 상의 다양한 component들 나타냄)
    - **ABAP Dispatcher** (중요, 빈출): 해당 SAP system 상의 교통정리(유한 자원의 할당 등). 역할에 맞게 요청을 process에 할당.
  - 하위 (5가지 work process 종류들 알아둘 것!)
    - **Dialog**(D) : 사용자가 사용하는 제시 화면, 리소스 많이 듬, 유한 자원, 여럿이 공유함. **Dispatcher당 최소 2개**.
    - **Update**(V) : DB에 대량의 Data를 update, 생성, delete 등을 수행하는 process. 시스템당 최소 하나.
    - **Background**(B): 사용자의 눈에 보이지 않게 처리되는 일들 (scheduling). Dispatcher당 최소 하나.
    - **Lock Management**(E = Enqueue work prosess) : system 당 한 개만 존재!!!! 동시에 data 접근을 막기 위함
    - **Spool**(S): 출력에 관련된 process 관리. 최소 시스템당 하나의 process가 요구됨. Dispatcher 하나 당 여러 개의 spool process 도 있을 수 있다.
- SM51: AS ABAP 시스템의 instance들 overview화면 제공.
- SM50: AS ABAP 시스템에 로그온한 사람의 instance에 돌고 있는 work process의 overview

- AS JAVA Processes
  - 가장 상위
    - JAVA Message Service(MS): JAVA dispatcher들의 list와 server process들을 관리.
    - Java Enqueue Service(ES): lock 관리
    - Software Deployment Manager(SDM): Software 개발 관련 지원해주는 매니저. 개발한 program을 server에 넘겨주고, 개발 툴로 가져와서 편집, 활성화할 수 있도록 해줌.
  - 중간
    - JAVA Dispatcher: 하위 프로세스들 교통정리. Incoming request들을 server process들에 나눠줌.
  - 하위
    - Server process(S) : JAVA app을 실행. 요청사항을 처리. Dispatcher당 최소 하나의 server process가 존재. 최대 16개.

- 회사에서는 SAP system에 ABAP stack/JAVA stack/ABAP+JAVA stack으로 선택 사용할 수 있다.
- ABAP+JAVA stack은 'add-in installation'이라고 불리기도 한다. ABAP부터 깔고 JAVA를 추가적으로 깔기 때문.
- **JCo**(JAVA Connector): ABAP stack과 JAVA stack이 통신하기 위해 사용됨.

<AS ABAP processes>

- **Buffer(Application level에 존재)**
  - DB로부터 받은 data를 일단 buffering. I/O 횟수를 줄이려고.
  - Application server 별로 가지고 있다.
  - Buffer는 사용자의 요청에 대한 처리 속도를 빠르게 해준다.
- Task-handler
  - 한정된 dialog work process를 공유하여 사용하기 위한 handler.
  - Shared memory에 있는 context를 roll in(context 가져옴)/roll out(context 저장)
- ABAP은 OpenSQL을 사용한다.
  - OpenSQL: SAP에서 사용하는 SAP에 종속적인 SQL.
  - Oracle, mySQL 등 조금씩 상이한 SQL들을 같은 방법으로 사용하기 위함.
  - Database의 종류에 독립적이다.
- Native SQL의 단점. (사용은 가능)
  - Application level에서 **buffering 수행 불가**
  - DB의 종속성 -> DB system이 바뀌면 전체 코드를 재작성해야 한다.

- Application level의 중요한 요소들.
  - ABAP interpreter
  - DB interface **(중요!!)**
    - 역할
      - OpenSQL->Native SQL로 변환.
      - Data 버퍼링 (Application server의 SAP 버퍼에)
  - Local buffer: 맨 처음 로컬 버퍼에 있으면 DB까지 가지 않고 buffer의 data 리턴.

<Processing Dialog Requests>

- 하나의 프로그램 수행에는 여러 개의 work process가 사용된다.
- 하나의 프로그램의 여러 스크린을 실행하기 위해서는 dialog work process가 필요한데 이를 'work process multiplexing'이라고 한다.
  (화면 하나가 하나의 dialog work process. PBO~PAI 사이가 하나의 dialog work process 사용하는 기간.)
- 하나의 dialog work process는 여러 user/program에서 공유하여 사용한다. 즉, dedicated 되지 않고 여러 번 잡았다 release해가면서 사용한다.

<Transactional Processing in AS ABAP>

- **Transaction**: processing unit. 4가지 속성이 있음-**ACID**
  - Atomic: 원자성(all or nothing). 전체가 성공하거나, 전체 취소되거나.
  - Consistent: 일관성 (DB의 무결성 지켜야)

- Isolated: 독립성. 하나의 transaction의 시작~끝 과정에서는 commit 전에 다른 transaction에서 볼 수 없어야 함.
- Durable: 지속성. 한번 commit된 것은 영속성을 가져야 함.

- P.90 DB Transaction과 SAP transaction의 relationship
  - PBO~Screen~PAI가 하나의 DB transaction
  - 화면이 뜨면 Dialog work process는 손 놓음.
  - 하나의 SAP transaction(= ABAP transaction)은 하나 이상의 DB transaction으로 구성. 즉, ABAP을 이용해 짧은 DB Transactions(implicit commit이 일어나는 tx들)를 모아 하나(SAP LUW-logical unit of work)로 만들 수 있다.

<Lock Management>

- SAP에서의 data consistency를 보장하기 위한 시스템.
- Enqueue process가 메인 메모리에 상주해 있는 'Lock table'을 관리함으로써 Lock 기능 수행.
- Lock되어 있는 데이터는 다른 사용자가 DB수정 불가.
- Lock mode 종류
  - Write lock(lock mode **E**xclusive): 변경을 하기 위해 lock. Lock을 건 사용자가 여러 번 lock 걸면 축적이 됨(cumulated)
  - Read lock(**S**hared lock): 여러 사용자가 read lock을 걸 수 있다.
  - Exclusive lock(lock mode e**X**clusive noncumulative): 오직 한번만 lock걸 수 있음. Lock을 건 사용자도 재접근 불가
  - Optimistic lock(lock mode **O**ptimistic): read lock처럼 동작하다가 변경된 data을 save하려고 할 때는 write lock으로 동작.
- **SM12**: 현재 setting된 lock 내역 보여줌.

<Standalone Enqueue server/ABAP central services>

- 화면 별로 일어나는 DB Transaction들은 commit이 너무 짧고 자주 일어나므로, 할 일들을 'VB log table'에 쌓아뒀다가(DB에 안 날리고) '**commit work**'를 만나는 순간 update work processor에 던져줘서 쌓인 일들을 실행. 한꺼번에 작업이 일어남. 즉, LUW단위로 commit하도록 함.

< Print >

- SAP GUI에서 Print 명령을 내리면 Dialog processor가 TemSe에 Spool request를 날리고, Spool processor가 Output request를 하면 TemSe에 있는 data를 프린트한다.(TemSe: spool 요청이 들어오면 쌓아두는 임시 테이블)
- 사용자 자신이 요청한 output request는 t-code '**SP02**'로 확인 가능

< Background processing>

- Background work process: 스케쥴 테이블에 task를 예약해서 정해진 시간에 처리하도록 함.
- 사용자의 입력이 반드시 필요한 작업은 background 처리 불가.
- **SMX**(T-code): Background 작업을 보여줌

<Communication via the Gateway>

- Gateway(or gateway reader): SAP ERP system과 SAP가 아닌 다른 시스템 간의 communication이 가능하도록 해줌.
  - Communication 방법: **RFC, CPI-C**

<Processing Web Requests>

- ICM(Internet communication manager)를 통해 가능.
  - Web 환경으로 부터의 request는 ICM를 통해 ABAP/JAVA dispatcher에 전달됨.

4. Communication and Integration Technologies

Key point: cross-system, ALE(Application linking enabling), RFC(remote function call), BAPI(중요!), SOA(Service oriented architecture)

- ALE: plant maintenance, accounting, sales 등 분산된 business process들을 통합할 수 있는 일종의 기술.
- BAPI(Business Application Programming Interface): business object의 method들.

<Remote Function calls and BAPIs> : 4장에서 가장 중요!

- SAP에서 사용되는 interface 기술들
  - ALE: Application link enabling
  - **BAPI**: Business Application Programming Interface
  - CPI-C: Common Program Interface Communication
  - EDI: Electronic Data Interchange
  - HTTP
  - LU 6.2: Logical Unit Type 6.2
  - **RFC**: Remote Function call
    - 원격 시스템의 predefined function을 호출할 수 있게 해준다.
      - SAP 시스템끼리도, 이종 시스템과도 RFC 통해 통신
      - RFC 사용을 위해서는 두 시스템 간에 technical connection을 맺어야 한다.(서로 destination을 지정, connection을 맺음)
        이를 RFC connection 또는 RFC destination이라고 부른다.
        Ex) (Calling system의 프로그램 코드 - 원격지에 있는 XY를 호출하겠다는 뜻. Called system에는 XY 함수가 정의되어 있음)
        CALL FUNCTION XY
        DESTINATION DEST
        EXPORTING …
        IMPORTING …
      - **SM59**: 어떤 시스템을 연결해서 통신할지 **Entry를 등록**할 수 있는 곳 (위 샘플 코드의 'DEST'가 여기에 미리 정의되어 있어야 함.
      - 순서: SM59에 원격 시스템의 Entry 등록->Calling system의 RFC호출(Destination은 지정해줘야 함)-> 미리 작성되어있던 Called system의 함수가 호출됨(샘플코드의 XY함수)
    - RFC는 communication process, parameter 전달, error handling등을 수행한다.
  - OLE: Object Linking and Embedding
  - SMTP: Simple Mail Transfer Protocol
  - SOAP: Simple Object Access Protocol
  - TCP/IP: Transmission Control Protocol/Internet Protocol
  - XML: Extensible Markup Language
- BOR and BAPIs
  - BOR: Business Object Repository
    - BOR에는 BO(Business object들이 존재. (ex. EmployeeAbsence)
    - BO는 method들을 가지고 있는데 이를 BAPI(Business Application Programming Interface)라 함. (ex. EmployeeAbsence.Approve)
    - 다시 말해, BAPI는 SAP 시스템 상의 BO들의 method들을 가리킴.
    - 원격지에서는 BO가 가지고 있는 BAPI를 통해 SAP system의 특정 Function을 수행 가능
  - BAPI의 사용목적
    - Business process들을 연결(link). Ex) ALE를 이용할 때
    - SAP Business suite의 framework에서 다양한 solution을 통합할 때 SAP에 의해 사용됨(ex. SAP CRM과 SAP ERP간의 연결)
    - SAP 시스템을 Internet에 연결할 때
    - SAP Business workflow와 연결할 때(conjunction)
    - 외부 프로그램과 연결할 때

- BAPI는 다른 function module들과 마찬가지로 **Function Builder(t-code: SE37)**를 통해 만들고 테스트된다. 만들고 나서 BOR내의 BAPI로 정의될 뿐이다.

- Enterprise Services-Oriented Architecture(Enterprise SOA)
  - SAP R/2(Application layer와 DB가 묶여있다)와 SAP R/3(Presentation/Application/DB layer)의 주된 차이점.
    - Host-based system과 client-server-based system이라는 것.
  - Client-server 구조와 Enterprise SOA(ex. B2B or WebService)
    - Client-server 구조: 하나의 application 안에서 하나의 Business logic을 처리
    - Enterprise SOA: 여러 개의 시스템에 분산되어 있는 business process 들을 하나의 process로 묶어 하나의 application으로 제공 가능해진 환경.
  - Enterprise SOA의 특징: 가장 중요한 키워드는 '분산'!. System이 분산된 환경에서 가장 유용한 architecture
    - application이 보통 여러 시스템에 분산되어 구현되어 있다.
    - Application은 ABAP이나 JAVA로 만들어진다.
    - 보통 application이 자신만의 데이터베이스를 가지고 있지 않다.
    - 현재 존재하는 시스템 밖의 기능들(function)을 Enterprise SOA application에 집어넣을 수 있다.
  - Enterprise service는
    - 새로운 application을 기존의 system을 modify하지 않고 효과적으로 만들 수 있게 해준다.
    - Business process를 설정하는 데 있어 아주 유연하다
    - 여러 시스템들의 기능들을 사용할 수 있다.
- Web Services
  - SAP NetWeaver AS에서 사용하는 Web service 표준들
    - eXtensible Markup Language (XML)
    - Simple Object Access Protocol(SOAP)
    - Web Service Description Language(WSDL)
    - Universal Description, Discovery, and Integration(UDDI): 이것의 도움을 통해 서비스를 제공하고 제공받을 수 있다.
  - Web Service 시나리오
    1) Service publication: Web Service provider가 자기가 만든 서비스를 UDDI에 등록
    2) Service search: Web service user가 자신이 원하는 서비스가 있는지 UDDI를 검색
    3) Service execution: 원하는 서비스를 찾으면 사용자가 서비스를 사용하기 위한 정보가 들어있는 WSDL(Web Service Description Language)를 사용하여 서비스를 이용한다.

5. Flow of an ABAP Program

: SAP는 3-tier(Presentation/Application/DB layer) 환경에서 Enterprise SOA로 넘어가고 있는 상태임.

- 3-tier
  - Presentation view
    - SAP접근방법
      - SAP GUI
      - Enterprise Portal (SAP Product 이름임)
      - Web Browser
  - Application layer
    - ABAP Program이 Application server에서 실행/처리 됨
  - Database layer
    - 실제 ABAP 프로그램은 여기에 저장되어 있다.
- Flow
  - Presentation layer에서 user가 요청한 프로세스를
  - Application layer의 Application server가 DB에 저장되어 있는 ABAP program를 메모리 영역으로 로딩해서 실행시켜 처리
  - ABAP program은 필요한 data를 Database에서 얻어옴.

- Process flow of a program with selection screen and list
  - Presentation layer에서 사용자의 서비스 요청
  - Application layer에서 ABAP runtime system이 ABAP 프로그램을 구동시켜 서비스 제공
    - 요청된 program을 DB에서 ABAP runtime system의 메모리로 올림.
    - 프로그램이 실행되고 Selection screen이 보여짐
    - 사용자로부터 필요한 입력을 받고 ABAP Business logic을 처리함.
      - 만약 Business logic을 구현한 function(function module)이 이미 있다면 reuse한다.
        (run-time 시에 Repository에 저장되어 있는 function module을 로딩)
        Ex. DB에서 data를 취득/저장하는 로직은 자주 쓰므로 function module로 만들어놓고 reuse
    - 결과 화면을 최종적으로 사용자에게 보여줌(presentation layer)

6. Introduction to the ABAP Workbench

- Repository(= Dictionary): Client에 관계 없이 볼 수 있다. Program, data, 각종 information등이 들어 있다.
  - Repository는 application components들로 구성(FI, MM, CO, SD…)
  - Application component는 여러 개의 package들로 구성
  - Package 안에는 관련된 object들이 논리적인 기준에 의해 구분되어 들어가 있다.
- RIS(Repository Information System: '**SE84**' – Object Navigator실행): Repository의 object들을 찾아볼 수 있다. (repository의 search tool)
- SAP Application Hierarchy('**SE81**'): application component들을 볼 수 있다.

Working with the object Navigator

- Screen Painter(SE51) : screen의 설정
- Debugger
- Menu Painter(SE41) : User interface 설계 (메뉴 바, 스탠다드 툴바, application 툴바, function key setting 등)
- ABAP Dictionary(**SE11**): DB의 table 정의 edit, central data type의 edit 등
- ABAP Editor(**SE38**) : 소스코드 Edit. ABAP 프로그램 개발 가능.
- Function Builder(**SE37**): Function module 유지보수
- Class Builder(**SE24**): global class들과 interface들 유지보수

- Object Navigator(**SE80**): Screen layout 잘 보자. (p.157)
  Navigation Area(화면 왼쪽, 계층구조 object list)/ToolArea(소스코드 있는 부분)/Context Menu(마우스 오른쪽버튼), 풀스크린 아이콘 etc.

Developing Programs and Organizing Developments (개발 절차)

- SAP에는 크게 3가지 종류의 system(또는 server)가 있다.
  - DEV(개발 서버)/QAS(테스트서버)/PRD(운영 서버)
  - Change Request: 개발자가 수정/개발한 내역을 담아서 'transport' 기능을 통해 다른 시스템에 반영한다.
  - 좀더 자세히~~
    - repository object를 생성/수정(cf. 하나의 object는 하나의 package에만 assign 가능.)
    - package에 생성한 repository object를 붙이고
    - change request에 붙여줘야 한꺼번에 이관(transport)됨.
    - Program/Table ⊂Package⊂Application component

- 개발된 ABAP Program 이름은 반드시 'Z' 또는 'Y'로 시작한다. (약속)
- ABAP 프로그램 언어의 특징
  - Type을 지키고
  - 다국어를 지원
  - SQL access 가능
  - OOP 지원
  - 플랫폼에 독립적
  - 상위버전에 대해 양립할 수 있다(Is upward-compatible)
- ABAP syntax
  - Individual sentence(statement)로 되어 있다
  - Statement의 첫 word는 ABAP 키워드이다.
  - 각 statement는 '.'로 끝난다.
  - 원하는 대로 들여쓰기 가능
  - ABAP runtime system은 대소문자를 구분하지 않는다.
- Comment
  - 줄 전체 주석은 맨 앞에 '*' 사용
  - 문장 중간부터 주석은 큰 따옴표(" ") 사용.
- Activating Programs
  - 모든 object는 크게 2가지 버전이 있음. Saved/Activated.
  - Saved program: 개발/테스트를 위함.
  - Activated program: 최종적으로 사용될 버전. 모든 시스템에 transport(이관)됨.

Create transactions
- Transaction 만이 role-based menu나 즐겨찾기에 포함될 수 있다. 거기 program을 넣고 싶으면 프로그램을 나타내는 t-code를 만들어야한다.
- Transaction은 하나의 repository object이므로, 한 package에 assign해야 한다.

- 하나의 project는 change request에 연결되어 있는데, 팀원들이 맡은 모든 task들이 release되면 PL이 change request를 release한다.
  -> QAS/ PRS로 이관할 수 있는 상태가 된다.

7. Basic ABAP Language Elements

Data type: 스크린의 I/O 필드를 만들 때 필요 formal variable description.
Data object: 변수 또는 상수.
- Data type을 사용하는 경우
  - 스크린 상의 I/O 필드 만들 때
  - Data object의 정의(definition)
  - Subroutine, Method등의 interface의 parameter type을 지정하기 위해
- **ABAP Standard types**
  - Complete ABAP standard type: fixed-length. 사이즈를 지정해주지 않아도 되는 data type.
    - D (Date): 'YYYYMMDD' 포맷. 길이 8
    - T (Time): 'HHMMSS' 포맷. 길이 6
    - I (Integer) : 길이 4
    - F(Floating point number): 길이 8
    - STRING: dynamic length 문자열 (길이가 정해져 있지 않음. 예외적 케이스)
    - XSTRING: byte sequence(hexadecimal string) (길이가 정해져 있지 않음. 예외적 케이스)
  - Incomplete ABAP Standard types: 항상 length를 지정해 주어야 함.
    - C(character string)
    - N(numerical character string)
    - X(byte sequence, hexadecimal string)
    - P(packed number-소수점 있는 숫자-)

Local Data Types : 현재 작성 중인 프로그램 안에서만 사용할 수 있는 data type. 반대는 'Global type'(ABAP Dictionary 안에 선언된 타입).
- 'TYPES' : local data type을 선언하는 키워드.

  Ex) TYPES gty_p_type TYPE p LENGTH 3 DECIMALS 2. "정수형 자리 3자리. 소수점은 2자리까지 갖는다.
  DATA p_value TYPE gty_p_type. "p_value는 정수형 3자리,소수점 2자리까지 갖는 p타입 변수이다.

  Ex2) 구버전의 표현도 가능.
  TYPES gty_c_type(3) TYPE c. "character type 3자리.
  TYPES gty_p_type(3) TYPE p DECIMALS 2. "첫번째 예제와 같은 뜻. 정수 자리 3자리, 소수점 2자리.

Global Data types
- Global types in Dictionary(=Repository). 3가지 type이 있다.
  - **Data element** :Field type
  - **Structure** : structure type
  - **Table type** : internal table들을 위한 타입
- Cross-client! : 모든 사용자, 모든 program이 볼 수 있다.

Definition of variable data objects
- 선언하는 방법(2가지) : 'DATA'라는 키워드로 만듬.
  - DATA 변수명 TYPE type_name.
    - Type_name에는 local type, global type(Dictionary에 등록된), standard type(D,T,I …)가 올 수 있다.
  - DATA 변수명 LIKE another_variable_name.
    - 기존에 정의 되어 있던 변수(variable = data object)인 another_variable_name의 타입을 참조해서 만든다.
- 변수의 초기값 설정: 'VALUE'라는 키워드 사용.

  Ex) DATA number TYPE i VALUE 17.
  DATA number2 LIKE number. "초기값 17은 복사되지 않는다.

Literals, Constants and Text Symbols
- Literals
  - 숫자표현(Numeric Literals)
    - 양수: 123
    - 음수: -123
  - 문자열
    - String: 'Hello'
    - Decimal number: '123.45'
    - Floating point number: '123.45E01'
- Fixed Data Objects with Label
  - **Constants** : 상수를 선언할 때 키워드. **프로그램 내에서 절대 변경될 수 없다.**
  - 문법: CONSTANTS gc_myconst TYPE type_name VALUE {literal|IS INITIAL}.
    CONSTANTS gs_pi TYPE p LENGTH 1 DECIMAL 2 VALUE '3.14'.

- Text Symbol
  - ABAP 개발은 **다국어를 지원**한다. 이를 위해 **Text Symbol**을 사용
  - Text symbol은 소스코드 바깥, 그 프로그램의 repository object의 text pool에 저장된다.
  - Text symbol은 다음과 같이 identify 한다.
    - Three character alphanumeric ID xxx. (ex. TEXT-001)
    - '…'(xxx)형태. (ex. 'Manager'(001))
      - '…'의 문자열은 로그온한 언어의 그 해당 text symbol내용이어야 한다.
  - 참고 시스템 변수
    - SY-DATUM: 현재 날짜
    - SY-UNAME: 로그인 유저 명

Comparison: Local and Global Data types

- Local: 프로그램 내에 만듦. Technical attributes only.
- Global: Dictionary에 만듦. Technical and semantic attributes

Basic ABAP Statements

- 다음은 같은 의미
  - MOVE gd_var1 TO gd_var2
  - gd_vard2 = gd_var1
  - 만약 두 타입이 같지 않으면 type conflict가 일어나는 데, conversion rule이 존재하면 자동으로 type conversion이 일어난다.
- 'CLEAR': resets the contents of a data object. 초기값으로 돌린다.

Calculations and Arithmetic Expressions

- 다음은 같은 의미. (COMPUTE는 optional!)
  - COMPUTE gv_percentage = gv_ccc * 100 / gv_max.
  - gv_percentage = gv_ccc * 100 / gv_max.
- 산술 연산에 사용가능한 operation들.
  - +
  - -
  - *
  - /
  - ** (Exponentiation)
  - DIV : 정수 나누기. 나머지 없이
  - MOD

Conditional Branches and Logical Expressions : **괄호( ), { } 안 씀!**

IF gv_var > 0. *statements* ELSEIF gv_var = 0. *statements* ELSE. *statements* ENDIF.	CASE gv_carrid. WHEN 'AA' *statements* WHEN 'LH'. *statements* WHEN OTHERS. *statements* ENDCASE.

case문에 'break'가 없다!

- IF문에 다음 문장들도 가능
  - IF gv_carrid **IS NOT INITIAL**. "gv_carrid가 값을 갖고 있으면
  - IF gv_carrid **IS INITIAL**.
  - IF **(** gv_carrid = 'AA' **OR** gv_carrid = 'LH' **) AND** gv_fldata = sy_datum.
  - IF **NOT** (gv_carrid = 'AA' OR gv_carrid = 'LH') AND gv_fldata > sy_datum.

Loop

DO. *Statements* IF <abort_condition>. EXIT. ENDIF. ENDDO. *(loop counter: sy-index)* *최대한 쓰지 말아야.* *무한루프의 가능성 때문.*	DO n TIMES. *Statements* *ENDDO.* *(loop counter: sy-index)*	WHILE <condition> *Statements* ENDWHILE. *(loop counter: sy-index)*	SELECT … FROM <dbtab> … *Statements* ENDSELECT.	LOOP AT <internal table> … *Statements* ENDLOOP. *(loop counter: **sy-tabix**)*

System fields

System field	Meaning
Sy-mandt	Logon client (ex. 800)
Sy-uname	Logon name of the user
Sy-langu	Logon language of the user
Sy-datum	Local date of the ABAP system
Sy-uzeit	Local time of the ABAP system
Sy-tcode	Current transaction code
Sy-repid	Name of the current ABAP program
Sy-index	Loop counter at DO and WHILE loops
…	…

- sy-subrc!! (중요한 system field!)
  - SQL문 수행 후 결과에 대한 데이터를 가져옴.
  - 이것이 **0이면 성공적으로 SQL문이 수행되었음**을 의미.

Dialog Messages

- 개발한 프로그램의 사용자에게 dialog 메시지를 보내고 싶을 때 사용.
- 문법
  MESSAGE tnnn(message_class) [WITH v1 [v2] [v3] [v4]].

- 위 문법의 t에 들어갈 타입들

Type	Meaning	Dialog Behavior	Message appears in
i	Info message	Program continues after breakpoint	Modal dialog box(팝업)
s	Success message	Program continues without breakpoint	Status bar of next screen
w	Warning	Context-dependent	Status bar
e	Error	Context-dependent	Status bar
a	Termination	Program termited	Modal dialog box(팝업)
x	Short dump	Runtime error MESSAGE_TYPE_X is triggered	Short dump

Working with the ABAP Debugger

- runtime시의 program의 상태를 보고 싶을 때 debugging 필요.
  - 해당 프로그램 열어놓고 breakpoint 걸어가면서 디버깅
- 여러 screen이 있는 program 디버그 시 command field에 '/h'를 입력하거나 메뉴에서 System->Utilities->Debugging ABAP(또는 Debugging screen)을 선택.
- 디버그 화면에서 변수 더블클릭하면 해당 변수의 그 시점의 값 보여줌.
- 프로그램이 동작하면서 **특정값을 갖게 됐을 때 프로그램을 세우고 싶으면 'Watchpoint'** 사용.
  - Watchpoints are breakpoints that depend on the field content.
  - Watchpoint는 **최대 10개까지 사용 가능.**
- 만약 필드값을 디버깅 중에 변경하고 싶으면 해당 변수를 더블클릭해서 write모드로 바꾼 후 값 변경하고 실행도 가능.

8. Modularization (재사용하자~)
: Sub-routine, Function module!

- IDEA: 너무 긴 프로그램, 중복코드가 많은 프로그램들을 모듈을 만들어서 읽기 쉽고 유지보수가 쉽도록 만들어보자.

- Local Program의 modularization
  - **Subroutine** (= form units)(FORM~ ENDFORM)
  - **Methods in local class**
- Global Modularization : 모듈을 global화 시켜서 여러 프로그램에서 사용할 수 있게 하자.
  - Function modules that are organized in function groups: function 모듈을 사용하는데, 이는 function group으로 묶여있다. (이름: SAPL~, SAPM~)
  - Methods in global class

- Encapsulating Data: 데이터를 서로 감추고 다른 모듈이 데이터를 직접 못 바꾸게 하자.
  - reuse하는 unit들(global class, function group의 function module)은 caller의 data를 직접 바꿀 수 없다.
- Function 호출 코드

```
CALL FUNCTION 'FUNCTION이름'
  EXPORTING
  ...
  IMPORTING
  ...
```

- METHOD 호출 코드

```
CALL METHOD
  cl_class=>method
  EXPORTING
  ...
  IMPORTING
  ...
```

Modularization with subroutines

- Subroutine의 호출

```
PERFORM sub_routine_이름
```

- Subroutine의 정의

```
FORM sub_routine_이름.
...
ENDFORM.
```

Parameter Definition for Subroutines

- Global variable: 만약 main program에 global variable로 정의되어 있으면 그 data는 subroutine에서도 사용 가능. 그렇지만, global variable을 쓰게 되면 재사용 가능한 subroutine의 장점을 잃게 됨.
- Subroutine을 사용할 때 subroutine의 interface를 사용하여 parameter를 넘겨주는, actual parameter를 사용해야 한다.
  Parameter를 전달하는 Subroutine interface의 3가지 방법
  - Call by value: 전달된 값을 subroutine에서 복사해서 사용함. 대용량 데이터를 함부로 전달하면 안됨. (부하걸림)
  - Call by value and result: 성격은 call by value와 동일하나, 정상적으로 subroutine이 종료 시(regular end)에는 result return한다.
    - 비정상적인 종료 (2가지)
      - STOP statement에 의해 종료
      - Type E user message
  - Call by reference: 주소값이 전달되어 main program의 variable이 변경된다.
- 문법! : **'CHANGING' -> 리턴한다**는 뜻.
  - Subroutine 호출 : a, b, c 즉 전송되어지는 부분을 'actual parameter'라 함.

```
PERFORM xyz
USING
a
CHANGING
b
c
```

  - Subroutine 정의 부분 : f1,f2, f3 즉 subroutine에 정의된 부분을 'formal parameter'라 함. 이 parameter는 subroutine안에서만 볼 수 있다.(이를 '**shadow rule**'이라 부른다.)

```
FORM xyz
USING
  value (f1) TYPE type_name… "Call by value
CHANGING
  value(f2) TYPE type_name… "Call by value and result
  f3… TYPE type_name  "Call by reference. Value 키워드를 안 쓰면 자동으로..
ENDFORM.
```

Modularization with Function Modules

Working with Function modules

- Function group
  - Function module들을 포함하고 있음.
  - Interface를 통해 외부 시스템과 통신 가능.
  - Global data object, subroutines, screen등을 내부에 공유해서 사용
- Function module
  - 내부(외부에서 접근 불가)
    - Properties
    - Local data objects
    - Source code
  - 외부와의 Interface
    - Import parameters: 부른 넘이 값을 function module로 전달해줄 때 사용.
    - Export parameters(optional) : 부른 넘에게 값을 던져주기 위해 export해주는 것
    - Changing parameters: 부른 넘에게 받아 변경해서 돌려주는 parameter
    - Exceptions
      - Sy-subrc값 이용
- Function module의 호출
  - 문법: CALL FUNCTION 'FM이름'.
- Function module을 찾는 방법
  - Application related search
    - Application Hierarchy (SE81을 통해 카테고리화 되어 있는 tree 뒤져서 찾기)
  - Application-independent search
    - RIS(Repository Information System) 뒤져서 찾기 (SE84)
  - Program-related search
    - 프로그램에서 'CALL FUNCTION' statement 찾아보기
    - '/h' 실행해서 debug모드에서 'CALL FUNCTION'문장에 breakpoint 걸어보기
    - F1 눌러서 Technical information 참고. Screen number를 알아내어 더블클릭해서 그리로 이동, "Where used list in programs" 실행

Documentation and Testing

- Documentation 내용
  - Short text
  - Function description
  - Example of use
  - Notes
  - Parameter Description
  - Exception Description
- Function module을 호출하기 위한 코드 생성하기
  - Drag and drop
  - Pattern 버튼 눌러서 'call function' 선택, 부르고 싶은 function module이름 지정.

Exception handling

- Sy-subrc값 참고해서 에러핸들링

```
Case sy-subrc.
 When 0.
  ...
Endcase.
```

Create function groups

- SE80에서 만들 수 있다.
  - Object navigator에서 object list 중 Function group 선택
- Function group을 만든 후에 Function module을 만듬.
  - Function group 만들고 나서 object tree상의 function group에 context 메뉴 띄워 Create->Function module
  - Customer function module은 'Z_' 또는 'Y_'로 이름이 시작해야함.

Working with BAPIs (reusable component)

- SAP의 BOR(Business Object Repository)에는 BO(Business object)들이 있다.
  - 이들은 일종의 class
  - 각 object들은 BAPIs(Business Application Programming Interfaces)를 method로 가지고 있다.
  - BAPI는 일반적으로 BO의 기본적인 function들을 위해 존재한다. (ex. Object 만들기, object의 속성 가져오기/바꾸기 etc)
- Standard BAPIs
  - GetList : available object의 리스트를 리턴
  - GetDetail: object에 대한 상세 정보 return
  - Create/Change/Delete/Cancel: create/change and delete objects
  - AddItem, RemoveItem: add/remove subobjects (예를 들면, item이나 어떤 주문정보)
- **BAPI**(t-code): BAPI를 검색하고 찾을 수 있는 BAPI Explorer.
- BAPI function module의 특징
  - Remote capability: 원격 control이 가능해야함.
  - No dialogs: dialog가 없다.
  - No exceptions: exception이 없다.
  - **에러메세지를 parameter를 export하는 특별한 방법인 'RETURN'으로 처리한다. (structure나 internal table 형태로)**
- Naming rule: 'BAPI_<business_object_name>_<method_name>'
- BAPI는 외부 시스템과 interface를 하므로 포맷을 맞추기 위해 conversion 루틴이 필요할 수도 있다.
- BAPI 호출방법: function module을 호출하듯이 호출하면 됨

```
CALL FUNCTION 'BAPI_....'
  EXPORTING
  ...
  IMPORTING
  ...
  return = ...
```

Modularization with Methods of Global Classes

Classes, Attributes and Methods

- Global class를 사용함으로써 Modularization 구현 가능.
- Class : Attributes+Methods
- Object 만들기: 'CREATE OBJECT'
- **SE24**: Class builder t-code. **Global 클래스**를 제어하기 위한 화면 제공.
- Class builder에서 import parameter를 넣어주고 method들의 testing 가능.

Exception handling

- Handling classic exceptions
  Class method 정의할 때 exception 정보도 넣어줌
  cl_example_compute=>get_something
  EXPORTING
  ...
  IMPORTING
  ...
  EXCEPTIONS
  POWER_VALUE_TOO_HIGH = 1
  RESULT_VALUE_TOO_HIGH = 2.
- Handling class-based Exceptions : 예외 자체가 객체로 넘어오는 경우
  - TRY~CATCH~ENDTRY문을 써야함.

- Class의 method의 소스코드 보고 있을 때 화면 오른쪽 상단의 'Signature' 버튼 누르면 현재 보고 있는 method의 parameter 확인 가능.
- 'Signature': interface parameters and exceptions

Modularization with Methods of Local classes
: 특정 프로그램 안에서만 정의하고 사용함.

- Local class의 구성은 크게 2가지
  - 정의부분
    CLASS lcl_compute DEFINITION.
    PUBLIC SECTION.
    ...
    PRIVATE SECTION.
    ...
    ENDCLASS.
  - 구현부분
    CLASS lcl_copute IMPLEMENTATION.
    ...
    ENDCLASS.

Unit 9. Complex data objects
:structure, internal table!!!

- ABAP 프로그램은 크게 두 가지
  - Report program
  - Screen program(type M, module pool program, dialog program): DB에 어떤 작업을 할 때, transaction이 일어날 때 보통 씀.
- Structure 만들기
  - Declaration
    TYPES: BEGIN OF < struct_타입명 >,
    carrid TYPE s_carrid,
    carrname TYPE s_carrname,
    …(components list)…
    END OF < struct_타입명 >
  - Definition
    DATA <변수명> **TYPE** <struct_타입명>.
- Data type은
  - Locally 선언되어 있거나
  - Dictionary에 있거나.
- Transparent table: DB에 존재하는 table의 구조와 똑같이 생겨서 transparent.
- **MOVE-CORRESPONDING** A **TO** B: A의 component들을 B로 copy. 반드시 필드명이 같은 것만 복사된다.
- Debugging 모드에서는 structure의 data object의 내용을 확인할 수 있다. ('structures' tab에서.)

Using Internal table

- Structure가 n개 있으면 table!
- DB에서 필요한 data를 갖고 와서 internal table에 담아 사용한다. (매번 DB에 access할 필요 없어지도록)
- Internal table의 Attributes(3가지)
  - Line type : 해당 internal table을 구성하는 row(record)의 구조.
  - Key
  - Table type (**기출**!)
    - **Standard** (default. 가장 많이 사용). 주로 Index를 통한 접근(=row number). Key가 non-unique. 검색속도 가장 느림
    - **Sorted**: 데이터가 이미 정렬된 상태로 테이블에 들어가 있음. 주로 key를 통한 접근.
    - **Hashed** : Key가 unique. Key로만 access. 검색속도 가장 빠름.
- 두 가지 선언문 구분하기 (**기출**!)
  - **DATA** abc **TYPE** scarr. : table scarr을 참조해서 만든 abc는 scar table의 line type과 같은 모양의 **structure**형태의 변수
  - **DATA** gt_flights **TYPE** bc400_t_flights. : table type(bc400_t_flights)을 참조해서 만든 gt_flight은 internal table.
- Internal table 만들기 첫 번째 방법
  - Local table type 선언하기 : bc_400_s_flight은 line type.
    TYPES gty_t_flights
    TYPE STANDARD TABLE OF bc_400_s_flight
    WITH NON-UNIQUE KEY carrid connid fldate.
  - Internal table 선언
    DATA gt_flights TYPE gty_t_flights.
- Internal table 만들기 두 번째 방법
  - Line type 정의
    TYPES: **BEGIN OF** gty_s_type,
    …(components list)…
    END OF gty_s_type.
  - Internal table 선언
    DATA gt_itab **TYPE [STANDARD|SORTED|HASHED] TABLE OF** gty_s_type

WITH … KEY …

- Internal table 만드는 3가지 방법 정리!
  - DATA gt_itab TYPE <Table Type>. : table type은 ABAP Dictionary 상에 있는 것.
    - cf. DATA: abc type <table name>. : 이 table은 물리적 테이블. abc는 structure
  - DATA gt_itab **TYPE [STANDARD|SORTED|HASHED] TABLE OF** <Structure type> WITH … KEY …
  - DATA gt_itab TYPE TABLE OF <Structure type>.
    - 가장 생략된 form. Non-unique한 default-key를 가진 standard table의 선언.
- Internal table에 가능한 연산 (ACIDR-외기)
  - Append : **APPEND** gs **TO** gt_itab. "structure gs를 gt_itab의 맨 뒤에 추가해라. 단, gs와 gt_itab의 line type은 반드시 모양이 같아야.
  - Insert: **INSERT** gs **INTO TABLE** gt_itab <condition>. "특정 위치에 행 추가. (ex)index 2)
  - Read: **READ TABLE** gt_itab **INTO** gs <condition>. " 조건에 맞는 structure 찾아서 복사.
  - Change: **MODIFY TABLE** gt_itab **FROM** gs [<condition>]. "gs에 있는 값으로 해당 조건의 라인을 변경해라.
  - Delete: **DELETE** gt_itab <condition>. "조건에 맞는 라인을 삭제하라.
- Internal table type 자체는 읽을 수 없다. 하나씩 꺼내 work area(structure)를 통해 읽어야 한다.
  - Work area 선언하기
    - gs_flightinfo **LIKE LINE OF** gt_flightinfo. "gt_flightinfo는 internal table 이름. 해당 table의 line type대로 structure를 만듬
- Sy-tabix: LOOP문의 line counter를 갖고 있는 시스템 변수..
- Sorting and deleting content
  - SORT gt_flightinfo. "해당 internal table을 key를 가지고 정렬
  - SORT gt_flightinfo **BY** carrid. "해당 column을 가지고 정렬
  - SORT gt_flightinfo BY percentage DESCENDING
    BY carrid ASCENDING. "두 컬럼 가지고 정렬도 가능
  - REFRESH gt_flightinfo. "table의 행들이 다 지워짐. 메모리는 반납 안함.
  - FREE gt_flightinfo. "table의 행들이 다 지워지고 메모리도 반납.
  - CLEAR: <internal_table> "요즘은 이렇게 많이 씀. Refresh와 동일 효과.
- 'WITH HEADER LINE'
  - 코드 샘플: DATA itab TYPE TABLE OF gty WITH HEADER LINE
  - 테이블 이름(itab)과 동일한 이름의 work area가 생긴다.
  - 장점: 따로 work area를 선언할 필요가 없다.
  - 단점: 문장이 모호해진다. Ex) INSERT itab INDEX 2. " insert itab to itab index 2. 테이블명이 생략됨.

Unit 10. Data Modeling and Data Retrieval

- **Transparent table** (**중요**!)
  - **ABAP Dictionary에 만든다.**
  - DB 안의 table과 똑 같은 모양.
  - Dictionary에 만드는 transparent table은 Structure type이다. (table 아님)
  - Transparent table을 dictionary에 만들면 같은 이름의 DB table이 자동으로 DB에 생성된다.
- 'MANDT': Client값(로그인할 때 화면에 잇는 코드)이 들어있는 필드명. 반드시 key값에 들어가야함.
- Transparent table의 기술적인 구조
  - Transparent table은 structure임!!!
  - Transparent table의 모양을 참고해서 DB의 table이 생성됨.
  - Transparent table의 field는 Data element의 모양을 참고해서 만듦.
    - Data element는 의미 있는 정보를 Domain을 참고해서 만듦. (ex. 도착도시)
  - Domain: 해당 필드가 물리적으로 어떤 형태인지 들어있음. (ex. CHAR 3)
- **Transparent table은 'SE11'(ABAP Dictionary)에서 조회 가능**.

Reading Database tables

- Database interface의 역할 2가지
  - OpenSQL->NativeSQL
  - Table buffering
- DB의 Table 찾기: DB, Dictionary에 있는 것들
  - 특정 application component안에서 찾는 경우. Application Hierarchy(SE81)로 찾는다.
  - Program에서 찾는 경우.
    - SELECT문장을 찾아서 debug 모드에서 ('/h') 해당 select 문장에 breakpoint 걸어놓고 들어오는지 확인.
    - 관련 structure filed를 F1눌러서 technical information을 띄우고, "where-used-list in table fields" 버튼을 눌러서 Dictionary에서 해당 table을 어디에 썼는지 리스트를 출력해 확인한다.
- SELECT문 문법
  SELECT <fields> FROM <table> INTO <target> WEHRE <condition>.
- SELECT SINGLE: 테이블에서 1건의 데이터만 들고 올 때 쓴다.
  Ex) SELECT SINGLE * FROM scar INTO ls_scarr WHERE carrid = iv_car. "where 조건에 full key를 다 걸어줘서 로드를 줄이자.
- SELECT문이 데이터를 성공적으로 들고오면 sy-subrc값이 0.
- **sy-dbcnt**: SELECT문을 통해 가져온 데이터 row의 개수.
- CORRESPONDING FIELDS OF: db구조가 바뀌어도 코드 수정할 필요가 없다. **But 속도는 좀 더 느려질 수 있다**. 컬럼 순서가 일치하지 않으면 알아서 일치하는 필드 찾아 매핑해서 넣어줌.
  SELECT SINGLE seatmax seatsocc
  FROM sflight
  INTO CORRESPONDING FIELDS OF ls_flight
  WHERE carrid = iv_carrid
  AND connid = iv_connid
  AND fldata = iv_fldate .
- SELECT 문에서 internal table에 한꺼번에 데이터를 담아올 경우(Array fetch)는 ENDSELCT문 안 쓴다. (구분법: 'INTO TABLE')
- CLIENT SPICIFIED: 생략하면 내가 로그온한 client의 data 중에서만 select문 수행. Client를 조건으로 주고 싶으면 이 문장 써야함.
  SELECT * FROM spfli
  CLIENT SPICIFIED
  INTO …
  WHERE mandt = 402
  AND carrid = 'AA'.
- where절에 key값을 쓰면 기본적으로 key 값에 대해서는 primary index가 설정이 되어 있기 때문에 검색 속도가 상당히 빠르다.
  (primary index는 만들지 않아도 생김)
- 특정 필드가 검색 조건에 자주 나타나는 데 키값은 아니라면 secondary index를 만들어 쓴다! 속도 향상을 위해.
- SAP Table buffer : Application server마다 각각 존재. 속도를 위해 한번 검색했던 결과를 버퍼에 가지고 있다가 나중에 Query 들어 왔을 때 버퍼에서 먼저 찾아온다.

- Table join하는 2가지 방법
  - Database view
  - 프로그램에서 두 테이블을 직접 join하는 것.
    SELECT …
    FROM spfli **INNER JOIN** scar
    ON spfli~carrid = scarr~carrid.
    WHERE …

Authorization check
: SAP R/3 시스템에서 특정 data들을 권한이 없는 사람들이 접근하는 것을 막기 위해 구현하는 컨셉.

- 'AUTHORITY-CHECK'를 통해 확인, sy-subrc값이 0이면 권한이 있는 것.
  : 현재 사용자의 master record를 체크.
  AUTHORITY-CHECK
  OBJECT 'S_CARRID'
  ID 'CARRID' FIELD iv_carrid
  ID 'ACTVT' FIELD '03'. "S_CARRID의 ACTIVITY를 체크해서 조회권한(03)이 있는지 확인.

Unit 11. ABAP Open SQL

- Application logic optimize하기
  - 관련된 field들만 선택한다
  - Where condition을 잘 준다.
    : 가능한 key field를 순서대로 고려해서 써야 index의 덕을 봐서 속도가 빠름.
  - 관련된 테이블들을 bundling해서 한 번에 처리하도록 한다.
  - Internal table에 담아서 Buffering 처리해서 동일한 select 명령을 줄인다. (array fetch)
    SELECT * FROM sflight **INTO TABLE** itab
    WHERE carrid **IN** ('AA', 'UA').
  - SQL을 잘 써서 한 건씩 가져오지 말고 대량으로 한번에 data 갖고 오자(mass access)
  - SAP table buffering을 이용

- 처음부터 data를 갖고 올 때 UP TO를 써서 제약을 둠.
  SELECT * FROM sflight INTO wa
  UP TO 1000 ROWS
  WHERE CARRID = 'AA'.
  ENDSELECT.
- LEFT OUTER JOIN: 왼쪽 table의 데이터와 매핑되는 데이터가 오른쪽 table에 없어도 결과로 출력함.
- SELECT 구문의 WHERE 안에 SELECT 사용 가능(subquery)
- **FOR ALL ENTRIES**: internal table과 DB 간의 join. **itab의 데이터는 중복되지 않고, 정렬되어 있어야함.**
  - 주의: outer_itab이 record를 안 갖고 있으면 scustom table에서 full select를 하게 된다. 꼭 outer_itab이 레코드 갖고 있는지 먼저 체크해야!
  SELECT * FROM SCUSTOM INTO …
  FOR ALL ENTRIES IN outer_itab
  WHERE id = outer_itab-customid.
  …
  ENDSELECT

Unit 12. Selection screen : data filtering하기 위한 화면!

- Selection screen의 Input field 만들기 (2가지) : 화면 1000번은 시스템에 의해 자동으로 생김.
  - **PARAMETERS**를 이용해서 만드는 방법
  - **SELECT-OPTIONS**를 이용해서 만드는 방법

Dynamic Preassginment of the selection screen

- Selection screen 환경에서의 ABAP event순서 (**기출!**)
  - LOAD-OF-PROGRAM
  - INITIALIZATION
  - AT SELECTION-SCREEN OUTPUT -> 사용자에게 화면 던지기
  - AT SELECTION SCREEN -> 사용자의 입력에 대한 ACTION. 입력한 데 대한 체크 수행하고 싶으면 여기서함.
  - START OF SELECTION : 실행 버튼 누르고 프로그램 수행.

Designing the selection screen

- Frame 만들기 - 문법
  SELECTION-SCREEN BEGIN OF BLOCK <block_name>
  WITH FRAME WITH TITLE <text>.
  SELECTION-SCREEN END OF BLOCK <block_name>.
- 한 줄에 display하기 : 주의 - 각 객체가 겹치지 않도록 포지션을 잘 지정해줘야함.
  SELECTION-SCREEN:
  BEGIN OF LINE
  COMMENT pos(len) <text> [FOR FIELD <f>]
  POSITION pos
  END OF LINE.

Creating Additional Selection screens

- SELECTION-SCREEN
  BEGIN OF SCREEN 1100.
  PARAMETERS: …
  SELECTION SCREEN END OF SCREEN 1100.

  …

  CALL SELECTION-SCREEN 1100
  STARTING AT 5 5
  ENDING AT 50 10.

Tab Pages on the Selection screen

- 먼저 subscreen을 만들어야함.
- SELECTION-SCREEN BEGIN OF TABBED BLOCK <tab_block_name> FOR 5 LINES.
  SELECTION-SCREEN TAB(10) tab1 USER-COMMAND comm1 DEFAULT SCREEN 101.
  …

SELECTION-SCREEN END OF BLOCK <tab_block_name>.

- Tabstrip block은 prog, dynnr, activetab라는 속성값을 가짐. Dynnr에 tabstrip의 스크린 번호를 넣고, activetab에 해당 command를 넣어 default tab을 설정한다.

Input checks and Variants

- **Input check**
  - **AT SELECTION SCREEN에서 수행**. (**기출**!)
    - 문법 (so_dept: 화면에 붙어있는 객체 이름)
      AT SELECTION-SCREEN **ON** *so_dept*.
      IF(조건).
       MESSAGE e046(bc405).
      ENDIF.
  - Selection screen에서 F1/F4(General help/Input help)의 기능을 만드는 방법
    - 문법 (ON HELP~: F1의 기능, ON VALUE~: F4의 기능)
      AT SELECTION-SCREEN
      ON HELP-REQUEST FOR [<param>|<sel_opt>]
      ON VALUE-REQUEST FOR [<param>|<sel_opt-low>|<sel_opt-high>]

      Ex)
      AT SELECTION-SCREEN ON HELP-REQUEST FOR *so_dept*.
        CALL SCREEN 100 STARTING AT 30 03
                        ENDING AT 70 10.
  - Variants : 사용자가 입력했던 값을 저장해서 언제든지 불러 쓸 수 있게 하는 기능.
    - Variant를 저장하는 방법 (2가지)
      1. ABAP Editor(SE38)에서 프로그램명을 집어넣고 해당 variant를 일일이 설정 가능
      2. 실행중인 프로그램에서 값을 입력 후 저장 버튼 눌러서 'save as variant'기능으로 저장.

Selection screen: modifications ant runtime

Generating Pushbuttons on the Selection screen

- TABLES sscrfields. "selection screen에서 발행하는 이벤트 처리 위해서는 반드시 이 라인이 있어야함.

  *push button
  SELECTION-SCREEN PUSHBUTTON /pos_low(12) *det_on* USER-COMMAND on. "det_on: 변수. on: 발생할 이벤트 이름.

  *pushbutton text
  INITIALIZATION.
    det_on = "Show details'(s01).
- Sscrfields-ucomm에 발생한 이벤트 이름이 들어감.
  Code:
  AT SELECTION-SCREEN.
    CASE sscrfields-ucomm.
      WHEN 'ON'.
      ...
      WHEN ...
    ENDCASE.
- PBO(Process before output): 화면 display 바로 전
- PAI(Process after input): 화면 display 바로 후. 사용자의 input 처리.

- Header line이 있는 internal table에 초기값을 셋팅하고 싶으면, sign(I/E)/option(BT..GT..)/low/high 옵션을 채워주고 append.

Selection screen changes at runtime

- SCREEN-ACTIVE: 런타임시에 특정 selection field들을 보이거나 숨기고 싶을 때 사용하는 키워드. 0이면 hide 1이면 display.

Unit 13. Programs calls and data storage management

- Executable program 실행방식 (3가지)
  - SUBMIT 함수이름 : calling 함수는 종료
  - SUBMIT 함수이름 AND RETURN: called 함수 수행 후 돌아옴(selection screen은 안 보임. Called 프로그램의 실행 결과 화면만 보임. back하면 calling program의 화면.).
  - SUBMIT 함수이름 VIA SELECTION SCREEN AND RETURN: called 프로그램의 SELECTION SCREEN을 보여주고 리스트 출력. BACK 두 번 누르면 원래 호출한 프로그램으로 돌아옴. (LIST화면-> SELECTION SCREEN -> CALLING 프로그램)
- Calling a transaction
  - LEAVE TO TRANSACTION 'T-CODE' [AND SKIP FIRST SCREEN]. : 해당 TRANSACTION으로 완전히 이동.
  - CALL TRANSACTION 'T-CODE' [AND SKIP FIRST SCREEN]. : 해당 TRANSACTION 종료 후 호출한 프로그램으로 다시 돌아옴.

Management of working memory

- SAP 시스템의 메모리.
  - ABAP memory: 하나의 external session에서 internal session들끼리 공유. 하나의 external session에 internal session은 9개까지.
  - SAP 메모리: external session들끼리 통신할 수 있는 공간. External session은 최대 6개까지.
- Internal session: 하나의 프로그램이 도는 단위라고 생각해도 됨.
- Main program이 특정 function module이나 function group을 호출하면 새로운 internal session이 올라오는 게 아니라 동일한 internal session에서 동작한다.
- 프로그램의 종류 (**중요**!)
  - Executable program(type 1) = report program
  - Module pool(type M)
  - Subroutine pool(type S)
  - Function group (type F)
  - Class pool(type K)

Data transfer : program A to program B (1 -> 5 갈수록 안 좋은 방법)

1. SAP memory를 통해 (SET/GET parameters)
2. ABAP memory (EXPORT/IMPORT, internal table: 'CALL TRANSACTION <T-CODE> USING bi_itab'. Bi_itab은 BDCDATA라는 structure를 가지고 있어야함.)
3. Interface: 해당 프로그램에 있는 스크린의 값을 이용할 수 있다. (SUBMIT ... WITH <스크린 필드값 가져오기> ...)
4. DB에 기록/읽기

5. File server에 기록/읽기

Unit 14. New Test tools
- Code inspector: ABAP Editor의 왼쪽 프로그램이 위치한 tree구조에서 context 메뉴 중 check->code inspector실행.
- 불필요한 변수 선언, 루프 체크 등을 코드 레벨에서 해서 결과를 알려줌.

<2주차 [TAW10_2-1, TAW10_2-2]>

SAP의 customer가 만드는 프로그램은 'Z'이나 'Y'로 시작해야 한다.
단, module pool program은 'SAPMZ'으로 시작한다.

Unit 1. Introduction to screen programming(= type M program. = module pool program. 용도는 DB update.)
- Naming rule: SAPM[Z|Y]이름 ex)SAPMSCREEN, SAPYEXAMLPLE
- program만들 때 'TOP include'에 체크하면 screen program.

General Aspects
- ABAP program types (5가지) : report, screen program만 자체적으로 실행 가능. 나머지는 다른 프로그램에 도움을 주는 프로그램의 유형임.
  - Executable program(Type 1) = report program : 직접 실행 가능.
  - Module pool program(type M) = screen program: t-code가 있어야 실행 가능.
  - Function group(type F)
  - Interface pool(type J)/class pool(type K)
  - Include program(type I)

Principles of Screen programming
- Screen 구성
  - Attribute
    - 종류: admin, type, size, sequence, settings
  - Layout
  - Flow logic

Creating screens (절차)
- Screen painter를 통해 그림.
  1. Screen attributes 지정
  2. Screen layout 그리기
  3. 각 screen element들의 속성을 줌
  4. Flow logic 구현.
- Screen number는 100단위로 -> 110, 120등 subscreen이 필요할 수 있기 때문.
- Include문의 이름
  - PBO: 'O01'로 끝나는 이름을 가진다.
  - PAI: 'A01' 로 끝나는 이름을 가진다.
  - TOP: 'TOP'으로 끝나면 global 변수를 가지는 부분.

Screen Modification and Sequence

- SCREEN system table: 현재 화면상의 모든 entry들에 대해 runtime시의 상태에 대한 정보를 가지고 있다.
- PBO가 불리기 전에 Design time시에 설계한 화면의 element들의 초기 속성을 Screen이라는 system table에 copy함.

Screen sequence
- Determining the next screen: static/dynamic 모두 가능.
- Dynamic screen sequences
  - Insert screen
  - Leave screen
- 'SET SCREEN 0': insert한 screen에서 원래 screen으로 돌아오는 코드. SCREEN 0이 존재하지는 않지만 약속된 표현임.
- 팝업 형태의 screen
  - Screen create시 속성을 'Modal dialog box'로 지정해줘야함.
- 'SET CURSOR FILED <필드이름> [OFFSET <o>].' : 동적으로 커서의 위치를 어떤 필드에 놓고 싶을 때 커서 위치 설정하기.

Unit 2. The program interface: program과 user간 interface

- Screen 프로그램에 title영역에 title을 설정하고 싶다면
  - Module pool program에 'SET TITLEBAR 'TITLE'.'이라 쓰고 TITLE 더블 클릭
  - Create object해서
  - Create title에 title값(화면에 display할 string값)을 넣어준다.
- Function key(F5, F8 등)는 이미 지정되어 있고 바꿀 수 없다. 사용할지 여부만 체크 가능.

Creating a GUI Status: 화면에 보여지는 거니까 PBO에서 만들어짐.
- SET PF-STATUS 'BASE'
- 'BASE' 더블클릭해서 create object.
- Short text, status type등 지정하여 저장.

Using GUI STATUS
- BACK 버튼에 대한 동작을 처리하고 싶다.
  1. MODULE user_command_100. : PAI에서 module문 호출
  2. Screen 설계시 element list에 타입이 'ok'인 element의 이름으로 ok_code를 설정해줌.
  3. DATA ok_code LIKE sy-ucomm.
     MODULE user_command_100 INPUT.
       CASE ok_code.
         WHEN 'BACK'
           LEAVE TO SCREEN 0.
       ENDCASE.
     ENDMODULE.

Unit 3. Screen Elements for Output
- Text field: text나 label. 보통 input field의 의미를 나타낼 때 많이 씀. 앞에 '_'. '?'를 쓰면 안됨. 다국어를 지원함.(multilingual)
- Status icons: 어떤 아이콘을 보여줄지는 runtime시에 결정된다. SE38: SHOWICON 실행. SAP에서 사용하는 모든 아이콘을 볼 수 있다.
- Group boxes: 관련된 여러 graphical element들을 박스로 묶을 수 있다. Title을 설정할 수도 있음.

Unit 4. Screen Elements for Input/Output

- Creating Input/Output Fields
  - 방법 2가지.
    - Screen painter를 이용하는 방법
    - ABAP Dictionary의 도움을 받는 방법.
- 화면 상의 Element 이름과 ABAP 프로그램 코드 상의 이름은 동일해야한다. (identical) 그래야 자동으로 data transfer 가능.
- Parameter id를 가지고 SAP memory를 이용해서 이전 데이터를 저장하고 가져올 수 있다. (같은 사용자가 사용했을 때) PAI processing block에서 SET parameter하고 PBO에서 get parameter하면 됨.

- Dialog message Categories(6가지. A/X/E/W/I/S **기출!**)
  - Termination
    - **A** message: 팝업창이 뜨고, screen 0로 돌아감. Authorization 관련 에러 처리시에 사용됨.
    - **X** message: 덤프화면으로 이동.
  - Error
    - **E** message : status bar 영역이 빨간색으로 바뀌면서 에러 메시지 출력
  - Warning
    - **W** message: status bar 영역에 노란색으로 메시지 출력
  - Information
    - **I** message: 팝업창. 이걸 쓸 때는 message 문 밑에 'STOP.'이라는 문장을 써줘야함.
  - Success
    - **S** message: 초록색으로 status bar영역에 출력됨.

- Automatic Field Input Checks (아래 순서대로 자동으로 input fields 체크됨)
  - Mandatory fields check : 필수키 필드인가 체크. (required인가)
  - Filed format check
  - Fixed values인가 check : fixed value – 해당 필드에 들어갈 수 있는 값이 Domain에 등록되어 있다.
  - Foreign key check.
- 추가적으로 input field를 check하고 싶으면 PAI에 logic을 구현한다.
- Checking Groups of fields: 여러 개의 입력 필드 체크 CHAIN.~FIELD: … MODULE check_input. ENDCHAIN.
- PAI모듈에서
  FIELD <field name>
  MODULE <module> **ON INPUT**.
  -> 화면상의 필드 이름을 프로그램에서 사용한 필드이름으로 찾아 초기값과 같지 않으면(is not initial) 해당 모듈 실행.
  여러 개 필드값 체크시에는 'ON CHAIN-INPUT'.
- PAI
  FIELD <field name>
  MODULE <module> **ON REQUEST**.
  -> 화면상의 값이 변경되었을 때만 실행. (사용자가 input 필드 수정했을 때.) 여러 개 필드값 체크시에는 'ON CHAIN-REQUEST'.
- 시스템의 자동 input field check를 피하는 방법
  - 주의사항: 'AT EXIT-COMMAND'를 사용/function의 타입을 type E로 주어야 한다.
  - 실제 소스 코드 : 의미- 사용자가 누른 버튼이 cancel이거나 exit일 때는 automatic check를 하지 말라. 대신 모듈문의 내용을 수행하라.
    - PAI 부분에서 function 호출

      ```
      MODULE exit AT EXIT-COMMAND.
      ```
    - 모듈 function 구현

      ```
      MODULE exit INPUT.
        CASE ok_code.
          WHEN 'CANCEL'.
            CLEAR ok_code.
            LEAVE TO SCREEN 0.
          WHEN 'EXIT'.
            LEAVE PROGRAM.
        ENDCASE.
      ENDMODULE.
      ```

Input help

- Input help = search help = possible entry=F4 help (F4 누르면 뜸)
- Drop-down boxes: 값 입력하는데에 drop-down list가 보이는 경우.
  - Drop-down input field가 나오기 위해서는
    - Search help가 있어야함 (F4 help)
    - 해당 data element와 연결된 domain이 fixed value를 가진 경우
    - 해당 field가 table을 갖는 경우. (20개 이하가 바람직)

Checkboxes, Radio Button Groups, and Pushbuttons

- Radio button: 여럿 중 하나 선택해야. 꼭 그룹으로 묶어줘야함.
- Check button: 다중 선택 가능.
- 체크된 상태 여부 확인은 c로 충분.
- Pushbutton
  - Layout editor로 끌어다놓기
  - Button의 function code 지정해주기: 버튼 클릭시 ok_code에 이 function code가 들어감.
  - Flow logic 구현

Unit 5. Screen Elements: Subscreens and Tabstrip controls

Subscreens

- Subscreen area를 만들고 여기에 어떤 subscreen을 끼워넣을지 결정하는 방식으로 구현됨.
- Subscreen의 사용 목적
  - Modularization of screens
  - Dynamic screen modifications
  - Reusable subscreens (만들어 놓은 subscreen을 다른 화면에서 재사용도 가능)

Processing subscreens

```
PROCESS BEFORE OUTPUT.
  CALL SUBSCREEN <subarea> "이 subscreen area에 include하겠다.
    INCLUDING <program name> <dynpro_number>. "program name에 있는 화면 번호가 dynpro_number인 subscreen을 포함하겠다.
```

PROCESS AFTER INPUT.
  CALL SUBSCREEN <subarea>.

- sy-cprog: 현재 실행중인 프로그램
- 다른 프로그램의 subscreen을 어떻게 불러다 쓸 것인가?
  답은 Function module(CALL FUNCTION)! 변수 등 data를 function module의 input parameter로 전달! (cf. function module도 일종의 프로그램임!)

Tabstrip controls

- 사용 목적: logically 관련된 정보를 각각의 탭별로 grouping. -> 사용자의 쉬운 데이터 접근 도모.
- Tabstrip control 만들기
  1. Tabstrip area만들기(끌어다놓기로 그림)
     - 소스: **CONTROLS:** my_tap_strip TYPE TABSTRIP. "주의! 'DATA:'가 아님!
  2. Tab title (버튼을 끌어다놓고 만듬.)
     - 버튼의 속성 중 **FctType** (**중요**!)
       - **P**: Local GUI function. Tab을 이동하더라도 PAI가 발생하지 않음. 'Scrolling locally'
       - ' ': 'initial'이라 부름. 의미는 Normal. 각각의 **tab을 누를 때마다 PAI이벤트가 발생**함. 'PAI scrolling'
  3. Subscreen area 구성 (각 탭은 subscreen들을 갖게 됨)

Scrolling locally in Tabstrip controls : Scrolling without triggering PAI – FctType을 'P'로 지정하자.

- Tab의 버튼을 클릭해도 화면을 갖고 오지 않음. 즉, PBO에서 모든 Subscreen 화면들을 가져왔어야 함.

```
PROCESS BEFORE OUTPUT.
  CALL SUBSCREEN subarea1 INCLUDING SY-CPROG '0101'.
  CALL SUBSCREEN subarea2 INCLUDING SY-CPROG '0102'.
  ···all subscreens···

PROCESS AFTER INPUT.
  CALL SUBSCREEN subarea1.
  CALL SUBSCREEN subarea2.
  ···all subscreens···
```

- PAI scrolling in Tabstrip controls : tab 클릭시 PAI이벤트 발생. 이벤트 발생시 해당하는 tab의 data의 내용을 가져옴.
  즉, 오직 사용자의 입력에 따라 어떤 subscreen에 어떤 화면을 출력할지 동적으로 결정한다.

```
PROCESS BEFORE OUTPUT.
  MODULE fill_dynnr. "display할 screen number가져와라.
  CALL SUBSCREEN subarea INCLUDING SY-CPROG dynnr.

PROCESS AFTER INPUT.
  CALL SUBSCREEN subarea.
  ···
  MODULE user_command.
```

Unit 6. Introduction to the Dictionary (SE11)

- ABAP Dictionary의 기능 (3가지)
  1. Type definition
     - Structure
     - Data Elements: structure를 구성
     - Table type: structure가 n개. Internal table의 구조를 describe.
  2. DB object 생성
     - DB Table
  3. Service 제공
     - Screen들
     - Possible entries.(F4 help)
     - F1 help
     - Etc.
- 모든 SAP 환경은 ABAP Dictionary를 바라보고 있다. (global하게 영향을 준다.)

Unit 7. Data objects in the ABAP Dictionary

Basic data types : data element, structure, table type, complex type: deep structure/nested structure

- SE11, ABAP Dictionary 실행 화면. 3가지 영역으로 구성.
  1. DB object를 만드는 옵션
     - Database table
     - View
  2. Data types
     - Data type
     - Type group
  3. Service
     - Domain
     - Search help
     - Lock object
- Domain: technical attributes를 정의함. CHAR, DEC, DATS, NUMC 등등. 직접적으로 사용할 수 없음. Data element를 통해야 함.
- Data elements: includes the semantic properties(의미적 정보)를 갖고 있다.
  1. Data element가 제공해주는 서비스들
     - Set/get parameters: SAP 메모리 사용
     - Search help
     - Field ID & translation(로그인한 언어에 맞게 label을 보여줌)
- Structures: Component들로 구성.
  1. Component가 될 수 있는 것들
     - Data element
     - View
     - DB table
     - Table type(itab)
  2. Nested structure: structure가 structure를 포함하는 형태. '-'를 통해 필요하면 여러 번 써서 데이터를 접근한다.
  3. Deep structure: structure 안에 table type이 존재하는 경우.
- Internal tables:임시적으로 프로그램에서 가공해 사용하는 테이블. DB에는 없다.
  1. 종류

- ◆ WITH HEADERLINE
- ◆ WITHOUT HEADERLINE

- Type group: global하게 사용할 constants를 정의하고 싶을 때 사용.
  1. Type group의 시작은 반드시 이런 형식: TYPE-POOL <type_pool_name>. ex) zmytp
  2. Type group 내의 constants들의 이름도 '<type_pool_name>_'를 prefix로 써야함. Ex) zmytp_pi

Tables in the ABAP Dictionary

Tables
- Column 구성
  - ■ Key fields: unique해야 함. 이를 통해 각 record를 정확히 선택할 수 있도록.
  - ■ Function fields: key field를 제외한 column

다국어 지원을 위해서는 master 테이블과 text table이 foreign key 관계가 설정되어 있어야 하고, text table에는 반드시 language key가 있어야 한다.
Maintenance view는 MV를 위한 Function view가 있어야 한다. Maintenance view를 통해 사용자가 화면을 통해 데이터를 변경/저장 가능.
Maintenance view: sm30에서 실행해볼 수 있다.

Search help를 달아주는 방법 두가지: program을 통해/ ABAP dictionary를 통해(better!)
화면 설계할 때 화면을 그리는 Dictionary상의 structure를 설계하고 structure에 search help를 attachment하는 방식이 가장 좋은 방법.

Unit 8. Performance when Accessing Tables
: 두 가지 관점에서 – index 사용 & Buffering

Improved Performance through Access per Index
- Index table: Select문의 조건에 index가 걸려있는 값이 주어지면, index table에서 binary search해서 데이터를 찾는다.
- **Optimizer**
  - ■ DB 레벨에 존재
  - ■ 최고 효율적인 방법을 판단하여 해당 데이터를 가져오도록 의사결정을 해줌. 이를 돕도록 index나 key를 잘 설정해줘야함.

Improved the performance through table buffering
- Select문을 수행할 때 Database interface를 통해 **SAP Table buffer**에 DB의 정보를 올려놓고 버퍼에 올려놓은 data는 DB가 아니라 buffer에서 가져옴.
  1. Full buffering: 테이블의 모든 data가 버퍼링된다. 버퍼의 사이즈가 제한적이기 때문에 무조건 full buffering은 바람직하지 않다.
  2. **Generic buffering(기출)** : key값이 조건에 일치하는 데이터들은 모두 버퍼링된다.
  3. Single-record buffering: 검색된 오직 하나의 record만 접근해서 버퍼로 가져온다. Single-record buffering을 쓸 경우 반드시 'SINGLE'을 select 문에 써야 한다.
     - ◆ SELECT **SINGLE** * FROM scounter
       WHERE carrid = 'LH' AND countnum = '00000004'.
- Synchronization
  - ■ 여러 AS(Application server)가 하나의 SAP 시스템을 공유하여 사용하기 때문에 DB와 buffer의 mismatching이 일어날 수 있으므로 sync가 필요함.
  - ■ Interval을 설정해놓고 주기적으로 sync를 수행한다.
    - ◆ Interval 짧으면 – system 부담 증가, 동기화는 개선
    - ◆ Interval 길면 – 시스템 부담 감소, 동기화는 worse.

Unit 9. Input checks
Value table/check table
Foreign key: 두 개의 table에서 자식 테이블의 column값이 부모 테이블의 column값을 참조하는 것을 가리킨다.

Input check via the technical domains
- Fixed values: 값이 한정적이고 바뀔 일이 거의 없는 것을 관리할 때 사용. Ex. Smoking/non-smoking, true/false …
- **Value table**: foreign key를 설정할 때 사용함. 즉, **시스템이 foreign key로 추천해줄 때 사용하는 테이블**) 어떠한 value check에 관여하지 않음..
  F4 기능과도 관련 없으며, 이 테이블에 있는 값만 사용 가능한 것 아님!!! 주의!!!
- Foreign key 설정을 잘 해놓으면, system이 알아서 input check를 어느 정도 수행함.
  - ■ Insert를 수행할 때, invalid input이 들어오면 foreign key관계가 있는 check table을 체크해서 에러 발생시킴. Insert도 안됨.
- 용어정리! (**중요**!)
  1. Foreign key table: Insert등을 수행할 테이블. (ex. SBOOK – 예약정보)
  2. Check Field: Foreign key관계를 맺을 Foreign key table의 field (ex. SBOOK의 counter)
  3. Check table: 체크할 값들을 갖고 있는 테이블. (ex. SCOUNTER – 존재하는 counter number를 갖고 있는 테이블)
- Foreign key relation을 맺어 놓으면 (= Foreign key 설정을 하면) Foreign key table에 insert를 하려고 할 때, check table에 가서 input check(=이상 유무)를 수행한다.

- 반드시 구분하자!!
  - ■ Value table: foreign key relation을 맺으려고 할 때 시스템이 제안해줄 수 있게끔 미리 설정해주는 table.
  - ■ Check table: 사용자의 data 일관성 체크를 위해 사용되는 테이블.
- **Check table과 value table은 다를 수도 있다!**

- **Cardinality**(Check table의 입장에서) : foreign key 설정할 때 셋팅값에 있다.
  1. **1:1**
  2. **1:N**
  3. **1:C** (1:1과 유사하나 check table의 영역이 mapping되지 않는 데이터가 있을 경우)
  4. **1:CN** (가장 일반적. check table의 영역이 mapping되지 않는 데이터가 있을 경우)

- **TEXT Table:** 로그온한 language로 text를 보여주고 싶을 때 Text table을 check table로 사용.
  - ■ **Language:** text table이 반드시 가지고 있어야 하는 key field값.
  - ■ **SAP 시스템은 다국어 시스템이다!**

Unit 10. Object Dependencies

- Active/Inactive: 모든 object에 대해서 두 버전이 양립함.
  1. Object를 만들고 activate하면 runtime object가 생김. 프로그램은 결국 이 runtime object를 사용하는 것.

- 'Where-used-list': SAP가 제공하는 기능. Ex) 언제 쓰나! - Domain을 변경하면서, 연관 있는 부분을 파악하고자 할 때
- RIS(Repository Information System) ABAP Dictionary: 일종의 Searching tool. ABAP Dictionary에 있는 object들을 특정한 조건에 따라 찾을 수 있다.

Unit 11. Changing Tables
: DB 안에 있는 Transparent table의 구조를 변경하고 싶다. 어떤 영향이 있을까?

- DB 안의 structure를 adjust하는 방법
  1. Delete& recreate: simple하지만, data가 없을 때에만 가능
  2. Change catalog on the DB(명령어: ALTER TABLE. 시스템이 허락해야 사용 가능.): 기존 table에 있던 index들은 다시 생성해야함.
  3. Convert the table: 가장 시간이 많이 소요됨.
     - ◆ 변경할 Table(A)에 lock을 걸어놓는다.
     - ◆ Table의 이름을 rename(A')하고 기존의 index를 삭제한다.
     - ◆ 빈 하나 더 만든다.(B)
     - ◆ Rename한 테이블의(A') 데이터를 B로 transporting한다.
     - ◆ Rename한 테이블은 DB에서 삭제하고, B테이블의 이름을 원래의 테이블 이름(A)으로 변경하여 저장, index도 새로 만든다.
     - ◆ Conversion 중 발생할 수 있는 문제점
       - Tablespace overflow: 특정 시점에서 중복된 데이터를 잡아놓고 쓰니까
       - Data loss if key is reduced in size: 중복된 키를 가진 데이터들은 자동 삭제되니까. (ex. 키값의 사이즈가 60->30으로 변경되면 30넘는 데이터는 잘림, 중복된 키값 가지게 될 수 있음. 데이터 삭제됨)
       - Invalid change of type: ex. Char -> int로 변경될 경우
     - ◆ Resuming Terminated conversions
       - Object log, dumps, system log등을 보고 재시행.
- Enhancement of SAP Standard tables
  - ■ SAP의 스탠다드 테이블에 두 칼럼을 추가하고 싶다.
    - ➔ Standard table의 수정 없이 append structure!
  - ■ Append structure의 이름은 'ZA'로 시작
    - ◆ Append structure의 field명은 'ZZ'이나 'YY'로 시작
  - ■ Append structure를 할 수 없는 경우 (**기출**!) (p.331)
    - ◆ Pooled/cluster tables
    - ◆ Table 안에 long field가 있는 경우(data type LCHR 또는 LRAW)
    - ◆ 필드 이름이 YY, ZZ으로 시작하지 않는 경우
    - ◆ Reserved namespace를 가지고 있는 경우. Namespace에 맞지 않으면 추가 append structure 사용 불가.

Unit 12. Views and Maintenance Dialogs

Restricted or Enhanced views on Database tables

- View: DB 안의 여러 개의 table에 흩어져 있는 data들을 하나의 view를 만들어 프로그램에서는 View를 통해 한 번에 접근할 수 있도록 함.
- View는 '**SE11**'에서 만들고 볼 수 있다.
- View의 종류 4가지
  1. Projection view
  2. Maintenance view
  3. Search help view
  4. Database view

- Join without 'ON' condition: cross-product가 생긴다. 반드시 JOIN할 때 조건을 달아줘야.
- Projection view: **하나의 table**에서 보고 싶은 필드만 선택해서 보고 싶을 때 사용.
- 두 개 이상의 table join할 때는 보고 싶은 필드 명 써줄 때 '~'로 어떤 table의 필드인지 지정해서 나열해줘야.
- SAP의 DB 2종류
  1. Database view: 물리적인 DB에 존재하는 view. Dictionary에 존재.
     - ◆ Dictionary 영역에서 view를 정의하고 activate하면 DB상에 view가 생성된다.
     - ◆ Dictionary상에 정의되어 있으므로 프로그램에서 type으로 선언해서 사용 가능. Dictionary상의 view 모양과 같은 structure형태가 됨. Ex) DATA wa TYPE zvo_myview.
     - ◆ Basis table(참초하는 실제 DB상의 테이블)에 대한 변경은(칼럼 추가/삭제) view 영역에도 자동으로 반영된다.
  2. Maintenance view: 일종의 **프로그램**. 유지보수를 위한 것.
     - ◆ DB layer에 존재하지 않음.
     - ◆ 사용자가 특정 table이나 data를 쉽게 넣고 뺄 수 있도록 해준다. 유지보수가 용이하도록.
     - ◆ 여러 개의 table 중에서 원하는 칼럼만 선택하여 view 생성. (projection view와 구분! Projection view는 단일 테이블에서!)
     - ◆ 1레벨 또는 2레벨의 스크린으로 구현할 수 있다. (화면이 제공된다. 열 편집 등이 가능)
     - ◆ Naming rule: 'ZV' 또는 'YV'로 시작.
     - ◆ **Maintenance view 생성 시 MANDT를 view field에 반드시 기입해야 한다.**
     - ◆ **SE11에서 만듦.**
     - ◆ **다른 view와 다르게 insert/modify등이 가능.**
     - ◆ **장점: simple maintenance dialogs can be built in a short period of time**
     - ◆ **단점: no asynchronous update. 비동기 업데이트를 지원하지 않음.**
     - ◆ **Maintenance view는 오직 Customizing과 customer tables를 위해 설계되었다.**
     - ◆ **View cluster(집합): 여러 개의 maintenance view를 foreign key relationship에 의해 논리적으로 엮을 수 있다. SE54에서 만들 수 있다.**
       - **장점**
         - ■ **Navigation : data에 손쉽게 접근 가능.**
         - ■ **Consistency: dependent한 모든 entries에 대해 일관성 체크를 자동으로 해준다.**

Unit 13. Search help (=F4 help = possible entry list제공)

- Search help도 ABAP dictionary 상의 object다. -> SE11로 생성 가능. 내부적으로 SELECT 문이 수행되서 table또는 view에서 데이터를 갖고 온다.
- Search help만드는 방법 (2가지)
  1. Dictionary상에 만들기
     - ◆ Dictionary 상에 Search help를 달 수 있는 곳들
       - Structure (가장 많이 사용)
       - Table
       - Data element
  2. 프로그램 상에 만들기
     - ◆ 코드
       - PROCESS ON VALUE-REQUEST
- **Collective Search help**

- **개념:** 사용자의 의도에 따라 각 Elementary search help를 만들고, 여러 Search help를 하나의 collective search help로 만들어 사용 가능. (cf. view cluster와 유사한 개념.)
- 고객의 욕에 따라 새로 구현된 search help를 standard search help에 끼워 넣을 수도 있다. (append search help)

- **Help view**(view as selection method of a search help)는 기본적으로 (left) outer join을 제공.

- Context 상 사용자의 입력 없이 input help 수행 시, 조건이 없는 SELECT문이 수행됨. 테이블 전체를 select할 것.
  ➔ System 부하 걸림. Input help에 대한 restriction 필요. (**search help exit**)
- Search help exit: SAPBC_GLOBAL_F4_SFLIGHT – 참고. Function module임. 'SE37'을 통해 모듈 내용 참조 가능.
  - SE37: Function module 조회/생성 가능.

- Input help 제공방법 3가지
  - Input help control(control 형태. Modeless)
  - R/3 Dialog(modal)
  - Listbox – key와 text 칼럼으로 구성되어진 경우에 listbox로 제공 가능.

<3주차 [TAW11_1]>

1. introduction

- ALV: ABAP List View의 약자. 일종의 class. Excel in place is a standard function(Excel로 데이터 다운받아 사용 가능. ALV의 중요기능)
- Include program: main program 안에 include program을 내포 가능.
- Classic ABAP list: 기존의 write문.
- ALV의 단점: **하나의 레코드에 multi-row display가 안됨**. End user에게 불편.

- 모듈 풀 프로그램(=Screen program = dialog program)의 Naming 룰: SAPM~ 또는 SAPMY
- Report 프로그램: Z~또는 Y~.

WRITE [AT] [/<pos(len)>] <field name> <opt1> <opt2>…
Ex) WRITE 2(4) 'ABAP'. => ABAP(두 칸 띄고 ABAP 출력)
WRITE 2(2) 'ABAP'. => AB(두 칸 띄고 AB 출력)
WRITE wa_sflight_price **CURRENCY** wa_sflight_currency. "만약 table에 들어있는 데이터가 100.10, USD이면 100.10 출력, 100.10, KRW이면 10,010 출력(환율 적용해서). SAP에서 통화의 단위는 소수점 둘째자리까지 표현 가능. TCURX 테이블에 해당 정보 들어있음.
WRITE … UNIT …: kg, cm등 단위에 관련된 write문 addition. T006테이블에 해당 정보 들어있음.

- Main program: 여러 가지 subprogram, include program들이 하나로 합쳐져 돌아가는 프로그램.
- ~F01: Subroutine들이 들어가 있다. (FORM ~ ENDFORM)
- ~ E01: ABAP event들이 들어가 있다. (START-OF-SELECTION 등)

2. Selection screen

Selection screen의 가장 큰 특징: PARAMETERS, SELECT-OPTIONS
- Selection program의 screen은 default로 **1000번부터** 만들어짐.
- Selection screen 호출 : CALL SELECTION-SCREEN 1100. cf) 일반 스크린: CALL SCREEN 100.

- PARAMETERS : 변수선언+selection screen을 만듦.
- SELECT-OPTIONS: 헤더라인이 있는 internal table을 선언해서 사용하는 변수문이다. (그니까 APPEND 등 사용 가능) Sign, Option, Low, High 필드가 있다.
  1. 문법: <seltab> FOR <f>
     Ex) SELECT-OPTIONS: so_car FOR wa_sflight-carrid DEFAULT 'AA'.
  2. Sign: I/E (I: 조건에 해당하는 data 갖고 옴. E반대. Exclude)
  3. Option: NB-not between, CP: contains pattern, NP(contains pattern not)

Dynamic Preassignment of the selection screen
- Selection screen 의 event 순서
  1. LOAD-OF-PROGRAM : 변수 초기값 설정
  2. INITIALIZATION: type 1 프로그램 only. (이벤트 프로그램. 실행 프로그램에서만 사용 가능)
  3. AT-SELECTION-SCREEN-OUTPUT(PBO): selection screen이 뜨기 전에 수행.
  4. AT-SELECTION-SCREEN(PAI): 에러가 있으면 E메시지화면, STOP명령에 의해 이전 화면 (PBO)로 돌아감. 주로 input check 수행.
  5. START-OF-SELECTION

Designing the selection screen
- 문법 : selection screen에 박스 만들기.
  SELECTION-SCREEN BEGIN OF BLOCK <block>
  WITH FRAME TITLE <text>
  Parameters… 또는 selec-options…
  SELECTION-SCREEN END OF BLOCK <block>
- 문법: 여러 element들 한 줄로 보여주기.
  SELECTION-SCREEN:
  BEGIN OF LINE
  COMMENT pos(len)<text> [FOR FIELD <f>] "화면 상의 INPUT FIELD 앞에 text 부분 지정 시 사용
  POSITION pos
  END OF LINE.
- Selection screen도 tabstrip 사용 가능.
  예제 프로그램:
  *Subscreens 만들기
  SELECTION-SCREEN BEGIN OF SCREEN 101 AS SUBSCREEN.
  SELECT-OPTIONS: …
  SELECTION-SCREEN END OF SCREEN 101.
  SELECTION-SCREEN BEGIN OF SCREEN 102 AS SUBSCREEN.
  SELECT-OPTIONS: …
  SELECTION-SCREEN END OF SCREEN 102.
  SELECTION-SCREEN BEGIN OF SCREEN 103 AS SUBSCREEN.
  SELECT-OPTIONS: …
  SELECTION-SCREEN END OF SCREEN 103.

  *만든 SUBSCREENS block으로 묶기

```
SELECTION-SCREEN BEGIN OF TABBED BLOCK tab_block   FOR 5 LINES "height of tab box
  SELECTION-SCREEN TAB (10) tab1 USER-COMMAND comm1 DEFAULT SCREEN 101. "10은 버튼에 들어갈 text의 최대길이
  SELECTION-SCREEN TAB (10) tab2 USER-COMMAND comm2 DEFAULT SCREEN 102.
  SELECTION-SCREEN TAB (10) tab3 USER-COMMAND comm3 DEFAULT SCREEN 103.
SELECTION-SCREEN END OF BLOCK tab_block.

INITIALIZATION
* tabstrip의 pushbutton 상의 text들 초기화
tab1= 'Connection' (001).
tab2= 'Flight' (002).
Tab3= 'Booking' (003).
* active tab 초기화
tab_block-activetab = 'COMM2'.
tab_block-dynnr = 102.
```

Input checks and Variants

- **AT-SELECTION-SCREEN에서 input check 수행함.** (**기출**!) (PAI와 비슷한 기능)
- 문법

```
AT SELECTION-SCREEN ON <field_name> ON <seltab> ON RADIOBUTTON GROUP <grp> On BLOCK <block>
```

- 예

```
AT SELECTION-SCREEN ON so_dept.
    IF ( so_dept-low LT '060000' AND so_dept-high LT '060000' )
      MESSAGE e46(bc405).
    ENDIF
```

- F1, F4의 기능을 사용자가 정의하고 싶을 때. (SAP에서 비추)

```
At SELECTION-SCREEN
ON HELP-REQUEST FOR [<param>|<sel_opt>] "F1 누를 때 technical information 대신 다른 화면으로 대체하겠다는 뜻
ON VALUE-REQUEST FOR [<param>|<sel_opt-low>|<sel_opt-high>] "F4 누를 때 다른 화면으로 대체
```

- 예

```
AT SELECTION-SCREEN ON HELP-REQUEST FOR so_dept.
  CALL SCREEN 100 STARTING AT 30 03
                  ENDING AT 70 10.
```

Variants

- Selection screen 상에 자주 입력하는 값들의 조합을 이름과 description을 지정해서 variant로 저장. 가져와서 사용할 수 있다.

Selection screen: Modifications at runtime

- Generating buttons on the selection screen

1. 문법

```
TABLES sscrfields. "명령어가 들어올 테이블 변수

SELECTION-SCREEN
    PUSHBUTTON /pos_low(12) det_on USER-COMMAND on. "on은 버튼을 눌렀을 때의 명령어.

INITIALIZATION.
  det_on = 'Show Details' (s01).

AT SELECTION-SCREEN. "PAI 역할
    CASE sscrfields-ucomm.
      WHEN 'ON'.
      ...
      WHEN ...
    ENDCASE.
```

Changing the selection screen at Runtime

- **'SCREEN'** : 동적으로 화면 상의 object들을 숨기거나 보여줄 때 사용하는 internal table.

1. SCREEN-ACTIVE : 0(숨김)/1(보여줌)
2. SCREEN-INPUT: 0(조회모드)/1(입력모드)

Unit 3. Logical Databases (= LDB). SAP만의 특징적인 데이터구조

- DB에서 Data를 꺼내는 방법

1. Open SQL : ABAP 프로그램 내에서.. DB 시스템에 독립적
2. Native SQL : DB system 의존적
3. LDB : SAP가 LDB를 만들어 놨으면 ABAP 프로그램에서 GET <node>해서 데이터 가져올 수 있다.
4. 기타(view…)

- LDB의 장점

1. Selection screen을 제공
2. Input/권한 체크 제공
3. 데이터 레코드를 읽을 수 있음.
4. 속도도 빠른편.

Subobjects and Data retrieval

- LDB 프로그램 예

NODES: spfli, sflight. "NODES를 쓰면 selection screen을 만들지 않아도 selection screen을 만든 효과. LDB에 관련된 table이름을 적어준다.

```
GET spfli FIELDS ... "DATA 취득
...
GET sflight FIELDS ... "SELECT를 안 쓰고 DATA를 가져올 수 있다. LOOP라고 생각하면 됨.
```

- LDB는 계층구조임. GET을 쓰면 그 계층적 구조를 반복적으로 LOOP수행해서 DATA를 들고온다. GET은 일종의 작은 이벤트.

1. START-OF-SELECTION ~ END-OF-SELECTION 안에 GET이 들어감.
2. Termination of an Event Block. 이벤트 블록에서 빠져나오는 방법 3가지.
   - ◆ CHECK: 제일 작은 범위, event block을 빠져나감. Ex) GET 이벤트 종료
   - ◆ STOP: CHECK보다 큰 범위. Event block을 빠져나가서 END-OF-SELECTION이 수행됨.

- EXIT: STOP보다 더 큰 범위. 리스트를 출력함.

- 예제로 사용하는 LDB, 'F1S'.
  1. 그 계층적 구조
     - SPFLI
       - 그 밑에 SFLIGHT
         - 그 밑에 SBOOK
- LDB programs
  1. Naming rule: **sapdb**<ldbname>
  2. 하는 일: DB access, input, authorization check.
  3. 들어 있는 내용: subroutine들의 collection.

```
FORM init.
...
ENDFORM init.

FORM put_spfli.
...
ENDFORM.
...
```

Unit 4. Data Retrieval without LDB
: GET 방식 말고 일반적으로 LDB에서 데이터 검색하기.

Reading Multiple Database Tables

- Database View in the ABAP Dictionary: 검색 전에 Dictionary에 미리 DB view를 만들어 놓고 사용한다.
  1. 장점
     - Central maintenance
     - Accessible to all users
     - Only one SELECT statement is required in the program.
  2. 단점
     - Low flexibility
- INNER JOIN :교집합 데이터만 갖고 옴.
  1. SELECT <필드명들> INTO TABLE <internal table> FROM 테이블1 **INNER JOIN** 테이블2 **ON …**
- OUTER JOIN: 왼쪽 테이블(테이블1)은 무조건 보여줌.
  1. SELECT <필드명들> INTO TABLE <internal table> FROM 테이블1 **LEFT OUTER JOIN** 테이블2 **ON …**
  2. SAP는 RIGHT OUTER JOIN은 지원 안됨.
- **FOR ALL ENTRIES** : DB에서 해당 조건을 만족하는 데이터를 모두 가져온다.
  1. 사용하는 internal table은 반드시 **sort되어 있어야** for all entries 실행 가능.
  2. **중복된 데이터는 제거해야한다.**
  1. **코드**

```
SORT itab_spfli.
DELETE ADJACENT DUPLICATES FROM itab_spfli.

SELECT carrid connid fldate…
  INTO TABLE itab_sflight   FROM sflight
  FOR ALL ENTRIES IN itab_spfli
  WHERE carrid = itab_spfli-carrid
  AND connid = itab_spfli-connid.
```

- Nested SELECT Statements: 가장 쓰지 말아야할 방법 중 하나. (퍼포먼스에 문제) SELECT 문 안에 SELECT.

Further Additions to SELECT statement

```
SELECT result "원하는 field들 나열
  FROM source "DB table이름
  INTO target
  [WHERE sql_cond] "검색조건
  [GROUP BY group]  "group level
  [HAVING group_cond] "어떤 조건으로 그루핑할 건지
  [ORDER BY sort_key] "DB에서 sorting해줌.
```

Ex1)

```
SELECT carrid connid
  SUM( seatmax ) SUM( seatocc ) COUNT( * )
  INTO TABLE it_sflight
  FROM sflight
  WHERE carrid IN so_car
  GROUP BY carrid connid.
```

- COUNT(DISTINCT connid) : 중복 제거.
- 정렬
  1. ORDER BY: DB에 로드 줌 default는 ASCENDING
  2. SORT: AS(Application server)에 로드 줌. AS단에서 정렬을 하려면 order by 대신 sort를 쓸 것.

Unit 5. Call programs with data transfer: 프로그램에서 다른 프로그램 호출 시 Data transfer 방법

- Running executable programs(type 1) : '**SUBMIT**'을 사용한다.
  가정: prog1에서 prog2를 호출한다.
  1. SUBMIT prog2. : 새로운 프로그램(prog2)으로 완전히 이동
  2. SUBMIT prog2 **AND RETURN**. : prog2수행하여 prog2의 결과화면(list)을 보여준 후 BACK버튼(F3) 누르면 prog1로 돌아간다. (AND RETURN 문 아래로)
  3. SUBMIT prog2 **VIA SELECTION SCREEN AND RETURN**: Prog2의 selection screen을 보여주고 난 후에 사용자 action에 따라 list를 보여주고 back 버튼 두 번 눌러야 prog1로 돌아감.
- Calling a transaction: t-code를 바로 실행 가능. '**LEAVE TO TRANSACTION**', '**CALL TRANSACTION**'을 사용한다.
  1. LEAVE TO TRANSACTION 'T_CODE' [AND SKIP FIRST SCREEN]. : 위의 2번과 같은 경우. T-CODE 수행 후 BACK하면 PROG1를 아예 수행하기 전으로 돌아감
  2. CALL TRANSACTION 'T_CODE' [AND SKIP FIRST SCREEN]: prog2에서 'LEAVE PROGRAM.' 또는 BACK하면 PROG1이 PROG2를 부른 바로 다음 라인으로 돌아감. (transaction을 so 프로그램으로 불러들이겠다는 뜻으로 기억해보자.)

Memory Management

- Logical memory model
  1. SAP GUI를 통해 맨 처음 로그인 하면 user session이 하나 열리는 것. (재로그인하면 또 다른 user session)
  2. 로그온해서 열리는 창 하나가 external session. '/o'로 창 하나 띄울 때마다 생김. 보통 최대 6개.
  3. External session에서 실행시키는 프로그램 하나하나가 internal session.
  4. Internal session끼리 공유하는 메모리가 ABAP memory
  5. Internal/external session 모두가 공유하는 메모리가 SAP memory. 범위가 넓은 대신 메모리 저장 공간은 작다.

Passing Data Between Programs

1. Interface: 예를 들면, call function, parameters
2. ABAP memory: internal session들끼리.
3. SAP Memory: SET/GET parameter. 같은 user session 안에서 공유하는 메모리.
4. DB: DB 안의 테이블을 읽고 쓰는 방법을 통해
5. 파일서버: 파일로 써서 (ex. 엑셀 파일, 텍스트파일…)
   ** 위 1번이 가장 좋고 5번으로 갈수록 안 좋다.

- Passing Data Using the program Interface
  1. Program과 Function group간 interface : IMPORTING/EXPORTING/TABLES/CHANGING/EXCEPTIONS 등 사용함
  2. Program들 간 interface: 'SUBMIT'문으로 selection screen의 input field들을 통해 데이터 전달. Ex) SUBMIT progB WITH pa_carr = 'AA'.

- T-code 만드는 transaction code: 'SE93'.
- ABAP Editor: 소스 만들기. 'SE38'.

- Passing parameters using ABAP Memory
  1. EXPORT … TO MEMORY ID 'MY_ID'. : ABAP 메모리 영역에 MY_ID라는 이름으로 데이터를 저장하자.
  2. IMPORT … FROM MEMORY ID 'MY_ID' : ABAP 메모리 영역의 MY_ID라는 곳의 데이터를 가져오겠다.

- Passing parameters using SAP memory : SAP 메모리는 하나의 ID당 하나의 변수만 사용 가능. **Work area, internal table 등은 불가.**
  1. SET PARAMETER ID 'CON' FIELD <변수이름>. : 'CON'이라는 이름으로 변수를 넣어놓는다. (전제조건: 모듈 풀 프로그램 스크린 그릴 때 'SET' 옵션에 체크되어 있어야 함.
  2. GET PARAMETER ID 'CON' FIELD <변수이름>. : SAP 메모리의 'CON'이라는 곳에 있는 값을 해당 변수에 넣겠다.

- Passing Data Using Internal tables: BDC program을 딸 때 많이 쓰는 방식.
  - BDC program: master data를 올릴 때 많이 쓰는 방법. BDCDATA에 하나의 프로그램을 자동으로 실행하는 부분을 기수해서 자동으로 프로그램을 실행시켜준다.
  - 코드

```
bi_itab TYPE TABLE OF bdcdata,
bi_wa TYPE bdcdata.

CALL TRANSACTION 'T_CODE'
        USING bi_tab
        MODE 'N'.
```

  - BDCDATA는 실행할 program의 이름, dynpro 스크린 number 등이 들어있는 internal table이다.

Unit 6. ALV Grid Control for Web Application 6.20 and lower

ALV: 'ABAP List Viewer'의 약자.

- **ALV를 만드는 순서**
  - Screen을 만든다. (Create screen 100번)
  - Screen painter에서 Control Area를 그려준다. (네모에 'c' 그려져있는 아이콘)
  - Source code에 Container control을 만든다. : cl_gui_custom_container
  - Enjoy SAP Control: ALV, tree등 graphic control을 채워넣음 :cl_gui_alv_grid
  - ALV 리스트를 display하기 위해 set_table_for_first_display라는 method를 호출한다.

- ALV 코드의 예

```
*data declarations
DATA: my_container TYPE REF TO cl_gui_custom_container, "container control을 위한 객체 변수 선언
      my_alv_grid TYPE REF TO cl_gui_alv_grid. "SAP control을 위한 객체변수 선언
*PBO of screen containing CONTROL_AREA
MODULE init_control_processing OUTPUT.
  IF my_container IS INITIAL. "screen이 호출될 때마다 object를 생성하지 않도록 조건을 달아준다.
  *  create container object and link to screen area
  CREATE OBJECT my_container
    EXPORTING
          container_name = 'MY_CONTROL_AREA'  "미리 만들어 둔 container 붙이기.
    EXCEPTIONS
          others = 1.
    IF sy-subrc NE 0.
          MESSAGE a… "error 메시지 처리
    ENDIF.

   *  create alv grid control abject and link to container
   CREATE OBJECT my_alv_grid
          EXPORTING
            I_parent = my_container  "grid와 container 연결
          EXCEPTIONS
            Others = 1.
    IF sy-subrc NE 0.
          MESSAGE a… "error 메시지 처리
    ENDIF.

  ENDIF.
```

- "Create object"를 SE38(ABAP Editor)의 pattern 버튼을 눌러서 할 수도 있다.
  - Pattern -> ABAP Object patterns 선택 -> Create object 선택. Instance 이름과 class 이름 입력. -> 실행.
- ABAP workbench(SE80)에서 해당 class를 찾아서 drag&drop 해서 Create object를 할 수도 있다.

- Releasing Control Instances
  - 객체는 메모리 구조. 프로그램 종료시 release 해줘야 한다.
    * free는 해당 class들의 method 중 하나. 객체와의 연결고리를 끊어준다
    CALL METHOD my_alv_grid->free.
    CALL METHOD my_container->free.

    FREE: my_alv_grid, my_container. "메모리 release.
- ALV의 ABAP program에서 해야할 일
  - List Data: 데이터 가져오기
    - List data를 가져오기 위해 Internal table이 있어야.
    - Global program data object나 class attribute를 써야한다. 즉, 글로벌 변수를 통해서만 ALV를 사용 가능.
  - Field catalog: ALV 화면에 어떻게 보이게 할 것인지, ALV의 컬럼의 속성을 제어함.
  - Additional info : 어떻게 정렬할 건지, filter 기능 등
- CALL METHOD my_alv_grid->set_table_for_first_display "DATA를 ALV에 던져서 화면에 보여줌.
  …
  - 예
    CALL METHOD my_alv_grid->set_table_for_first_display
      EXPORTING
        i_structure_name = 'SBOOK' " SBOOK의 structure 모양대로 화면을 뿌려라.
      CHANGING
        It_outtab = it_sbook "it_outtab: 데이터가 담긴 곳
      EXCEPTIONS
        OTHERS = 1.
- CALL METHOD 역시 pattern, SE80을 통한 drag&drop을 통해 코딩 가능.
- CALL METHOD my_alv_grid->**refresh_table_display** "ALV 리스트가 변경되었을 때 다시 읽어오겠다는 뜻. 조회용 프로그램에서는 안 써도 됨.
    EXPORTING..

ALV Grid control: Layout Variants
: ALV의 layout구성을 저장했다가 필요할 때 불러다 쓰자.
- Procedure

1) DATA: my_variant **TYPE disvariant.**
   PARAMETERS: pa_lv **TYPE disvariant-variant.**

2) my_variant-report = sy-prog. "현재 프로그램 이름.
   my_variant-variant = pa_lv. "parameter가 받은 variant 이름.

3) CALL METHOD my_alv_gruid->set_table_for_first_display
     EXPORTING
     …
     is_variant = my_variant
     …

ALV Grid control: Layout
- Structure type, '**lvc_s_layo**' : layout의 형태를 조절할 수 있게 해주는 structure
  - Ex: grid title, zebra, no_headers sel_mode, …
- Light, cell 색깔 등 조작 가능.

ALV Grid control: Field catalog : end user의 화면에 보이는 내용, internal table의 내용 중 보여줄 내용, 정렬방법 등에 대한 지정을 field catalog를 통해 한다.
- LVC_T_CAT: column의 properties를 갖고 있는 global data type. (table) row type은 'LVC_S_CAT'.
- set_table_for_first_display 메소드 호출 시 field catalog(IT_FIELDCATALOG)와 structure(I_STRUCTURE_NAME) 중 하나는 반드시 필요하다. 없으면 에러! 동일한 칼럼 존재 시 카탈로그가 우선.
- Formatting amounts appropriately for units
  - CFIELDNAME : column name for currency formatting
  - CURRENCY: explicit currency abbreviation for an entire column
  - QFIELDNAME: column name for unit formatting
  - QUANTITY: Explicit unit abbreviation for an entire column.

ALV Grid control: Events
- Control framework Architecture를 통해 event처리를 해줌.
  1. 사용자가 dialog 화면에서 이벤트를 발생시킴. (Presentation server에서)
  2. Automation controller가 반응 (이후는 Application 서버에서)
  3. CFW호출
  4. Representative instance가 반응.
  5. Basis services 뒤져서 발생한 이벤트에 맞는 반응을 보임.
- 관련 class: cl_gui_alv_grid. : 궁금하면 SE24(class builder)로 조회 가능. Event tab눌러서..
- ALV Grid control events Through mouse operations(**기출**!!)
  1. DOUBLE CLICK
     - Export parameters:
       - ES_ROW_NO: 더블클릭한 라인의 라인 no.
       - E_COLUMN: 더블클릭한 칼럼이 어디? 필드 이름 알려줌.
  2. HOTSPOT_CLOCK: 마우스 갖다 댔을 때 손가락 모양이 됨.
     - Export parameters
       - ES_ROW_NO
       - E_COULUMN_ID
- ALV list 안에 버튼을 만들어 넣을 수도 있다.

Unit7. Appendix: Background Processing
SM37: runtime시에 돌고 있는 Job overview

<3주차 [TAW11_2]>

Unit 1. Database Updates with Open SQL

- Introduction
  - Open SQL: single record access하지 말고 set access! 즉 internal table에 담아 가져오자. Performance 때문!

- 관련 system 변수
  - sy-subrc : 0이면 일치하는 데이터가 한 건이라도 있다.
  - sy-dbcnt: 결과 값의 수. 1건이 올라왔으면 sy-dbcnt는 1.
- 'CLIENT SPECIFIED': select문이나 open sql문에 client를 조건으로 써야하는 경우 이 명령 사용하면 조건에 'MNDT' 사용 가능

Open SQL: Syntax

- INSERT <dbtab> [CLIENT SPECIFIED] FROM <work area>. "work area에 데이터 채워서 dbtab(db table)에 해당 work area를 추가. **(single row)**
- INSERT <dbtab> [CLIENT SPECIFIED] FROM TABLE <internal table>.
  - internal table에 데이터 채워서 dbtab(db table)에 **한꺼번에 여러 row를 추가**.
  - 만약 한 건의 데이터라도 문제가 생기면 전체 데이터가 안 들어감 (all or nothing)
  - 만약 중복 데이터도 허용해서 insert 하고 싶으면 'ACCEPTING DUPLICATE KEYS'를 추가해주면 됨.

- UPDATE <dbtab> [CLIENT SPECIFIED] FROM <work area>. "UPDATE: 데이터를 변경하고 싶을 때.
- UPDATE <dbtab> [CLIENT SPECIFIED]
  SET <f1> = <g1> … <fn> = <gn> "field(fx)값을 변수값(gx)으로 바꿔라.
  WHERE <full qualified key>. "조건에 맞는 값만.
  - 만약 WHERE절의 조건이 키 필드 전체에 조건을 걸어주지 않으면 single record만 바뀌는 게 아니다. 키값의 일부만 where 절에 있으면 여러 건이 변경되게 된다.
- UPDATE <dbtab> [CLIENT SPECIFIED] FROM TABLE <internal table>.

- MODIFY <dbtab> [CLIENT SPECIFIED] FROM <work area>. "Single record 변경
- MODIFY <dbtab> [CLIENT SPECIFIED] FROM TABLE <internal table>. "multi row 변경
- MODIFY의 의미
  - UPDATE & INSERT: 해당 데이터가 이미 존재하면 업데이트하고, 아니면 INSERT.
  - Sy-subrc가 0일 확률이 거의 없음. 0이면 DB system이상이거나, table만들 때 initial 체크가 안 되어 있을 때.
  - **Internal table의 modify와 조금 다름.** Internal table의 modify는 internal table의 data를 변경하는 것.

- DELETE FROM <dbtab> [CLIENT SPECIFIED] WHERE <full qualified key>. "single record 삭제
- DELETE FROM <dbtab> [CLIENT SPECIFIED] WHERE <condition>. "multi record 삭제도 가능.
  - 주의!: WHERE 조건을 안 쓰면 table상의 모든 레코드가 삭제됨.

- Restoring previous database status
  IF sy-subrc NE 0.
  MESSAGE A… "abort! "roolback work" 기능을 한다. 실패하면 수행했던 것을 취소해라.
  ENDIF.

- SAP에서는 commit, rollback을 이렇게 쓴다.
  - "Commit work."
  - "Rollback work."

Unit 2. LUWs and Client/Server Architecture
: LUW 두 종류 – SAP LUW/Database LUW. 논리적인 연관성이 있는 내용을 묶어서 작업하도록 한다. (Logical unit of work)
- Database LUW: program상에서 'rollback work' 또는 'commit work'(='sealed')를 만났을 때 까지가 DB LUW.
- SAP LUW
  - 여러 번의 DB commit이 implicit하게 일어난다. (PBO~ PAI)
  - 의미적으로 하나의 묶음이 SAP LUW가 될 것. (V1/V2를 공부하자.)

Unit 3. SAP Locking Concept
:DB에서 lock을 처리하는 것이 아니라 SAP에서 처리한다.
Lock을 하는 이유? 같은 Data에 동시 접근을 막고 싶다. (avoiding concurrent access to the same data)

The SAP locking concept.
- **Database가**(SAP가 아님) 스크린에서 빠져나올 때마다 DB-commit -> DB-commit은 DB lock을 풀어버린다.
- **Enqeue work process가 Lock table을 관리하면서 lock 관리. Lock process(lock table)를 보려면 t-code, 'SM12'.**

Lock Objects and Lock Modules

SAP Lock Objects
- SAP lock object를 만들 때는 "EZ" 또는 "EY"로 시작.
- ABAP Dictionary에 만듦(**SE11**)

Generating lock modules
1) Definition: lock을 걸 object를 정의한다.
2) Activate: 해당 object를 Activate한다. -> Function이 자동으로 2가지가 생김. ENQEUE_object/DEQEUE_object. (for setting locks/for releasing locks) 프로그램에서 lock을 잡거나 풀 때 이 function을 호출하게 된다. 파라미터는 full key.

Setting and releasing locks

Setting, Removing and administering SAP locks
- ABAP program이 lock module을 호출(ENQEUE~ 또는 DEQEUE~). -> LOCK 모듈이 LOCK TABLE을 통해 LOCK할 데이터를 등록/삭제.
- ENQEUE, DEQEUE 호출 시 파라미터는 반드시 동일해야함.
  코드: CALL FUNCTION 'ENQEUE_ESFLIGHT'
  EXPORTING
  CARRID = …
  CONNID = …
  FLDATE = …
  …
- 만약 LOCK이 실패했다면, 이유는 2가지.
  - Entry already blocked(Exception '**FOREIGN_LOCK**') – 다른 넘이 이미 lock 잡음.
  - Error in lock management (Exception '**SYSTEM_FAILURE**') – 시스템 이상.

- Lock argument: "SM12"로 볼 수 있다. 아래는 볼 수 있는 정보들.
  - Call parameters (client 정보 포함)

- Lock argument : 시스템이 어떤 lock을 걸고 있는지 알 수 있음. (client 정보 포함. ####있으면 multi로 lock 잡고 있는 것. 전체가 숫자는 single record lock.)
- Lock effect

- ENQUEUE 모듈의 parameters
  - Mode_<tabName>
    - E: cumulative write lock . 같은 사용자는 여러 번 LOCK잡을 수 있지만, 다른 사용자는 이미 LOCK걸려있다고 에러.
    - X: non-cumulative write lock. 한 번 LOCK걸리면 아무도 다시 걸지 못함.
    - S: cumulative **read lock :**LOCK걸려 있어도 다른 사람들이 다시 걸 수 있음. X LOCK만 못 건다.
  - _wait: 'X'또는 ' '. Lock이 실패했을 때 시스템이 정한 시간동안 기다렸다 재시도를할지 옵션.
  - _collect: 'X'또는 ' '. Lock을 동시에 여러 개 잡을 때 한꺼번에 잡아야한다는 뜻. Function module '**FLUSH_ENQEUE**'을 호출해야 CONTAINER에 담겨있던 모듈들이 LOCK이 걸림. **한꺼번에 이 LOCK들 풀 때는 'DEQEUE ALL'**.

- 올바른 lock 사용 순서: lock->read->change->release

Unit 4. Organizing Database updates
V1/V2!?

Database changes from within the application program.

- Timescale: 가장 일반적인 방법.
  - 프로그램의 각 screen 마다 DB update를 수행하지 말고, 마지막 dialog 의 동작이 끝났을 때 한꺼번에 처리한다.
- Data flow
  - 모든 dialog에서 사용하는 data를 TOP영역에 선언하고, 마지막 dialog의 동작이 끝났을 때 DB에 저장한다.
  - TOP에 선언된 Global program data는 마지막에 한 번만 update할 거니까 update할 data들을 global 영역에 모두 저장해 놓아야 함. 그리고 그 값들을 중간에 변경하지 않아야 함.
- 위의 방법의 문제: lock duration이 너무 길 수 있다.

Using **delayed subroutines** for Database Updates

- **POC** timescale(PERFORM ON COMMIT)**: Parameter 사용 불가.** ex) PERFORM x USING A (X)
  - 각 dialog에서 POC를 system table에 등록만 해놓다가 마지막 dialog 수행을 마치고 'COMMIT WORK'를 통해 쌓여있던 PCO들을 수행하게 됨.
  - 코드

```
PERFORM x ON COMMIT.
…
PERFORM y ON COMMIT.
…
COMMIT WORK.
*    x,y는 subroutine이다.
FORM x.
        UPDATE tab1…
ENDFORM.
…
FORM y.
        UPDATE tab2…
ENDFORM.
```

- POC의 특징 (FORM~ ENDFORM)
  - POC는 parameter를 사용할 수 없다.
  - 'Commit work'를 만나야 실행된다. (delayed subroutine임)
  - 이들은 global data를 사용한다.
  - **'Rollback work' 대신 message A type을 사용한다.**
- POC: 재사용의 의미가 아님. 각각 따로 만들어줘야함. (save, save1…)

Database changes using update techniques

- DB technique이 필요한 경우
  - 사용자가 변경사항이 완료될 때까지 한참 기다려야하는 경우
  - Dialog work process가 release되지 않는 경우
  - Error logging이 안 되는 경우
  - 여러 DB work process가 동시에 일어나는 경우 -> DB 퍼포먼스 저하.
  - LUW가 Locking concept 쓸 때 system function으로는 지원이 안됨. 프로그램 내에서 구현되어야함.
- Update 테크닉의 기본 흐름.
  - Request를 바로 DB에 기록하지 않고 log table에 갖고 있는다. 하나의 log table이 하나의 DB LUW라 보면 됨. (POC와 유사)
  - Commit work 만나면 log table의 내용을 DB에 반영
    - DB에 반영 성공하면 lob table 삭제
    - 실패하면
      - ROLLBACK WORK 수행
      - 또는 MESSAGE A도 로그 테이블 삭제.

Technical Implementation of the update

- Function module을 통한 DB 업데이트.
- 들어오는 Data는 있으나, 나가는 데이터는 없다.
  - **Exporting, changing, tables는 사용할 수 없다.**
  - **Importing, exceptions만 사용 가능.**
- Log table과 마찬가지로 message A가 rollback work의 기능을 함.
- Function module은 SE37에서 만든다.
- **POC는 parameter 사용 불가. Global 변수만 사용. But function module은 parameter 사용 가능**
- 'CFIUT'라고도 함. (약자.)
  - 의미는 '바로 실행하지 말고 로그 테이블에 쌓아놓아라.
  - 코드: CALL FUNCTION <function name> IN UPDATE TASK.
  - CFIUT 여러 번 호출 후, 'COMMIT WORK' 불러야 DB에 반영됨.
- Log table = VBLOG라고 부르기도함.
- Rollback work or message 'A'
  - Log table에 있는 request 전체 삭제
  - 미리 잡혀있던 lock 전체 삭제
  - 현재 LUW 안에서의 모든 변경사항 취소

- Perform on commit(POC)으로 등록된 subroutine 삭제
- Log table의 내용을 삭제하는 경우 3가지
  - Commit 성공
  - Rollback work
  - Message 'A' : 팝업창 뜨고, 작업 취소 가능.
- **UPDATE 모듈의 function 내에서는 rollback work/commit work 사용 불가. Message A만 사용 가능.**
- Setting locks in the update
  - Locking using _scope = 2: lock이 V1까지는 pass가 된다. (1일 때는 transfer되지 않음)

Use and effect of the update modes
- Asynchronous Update
  - LUW1과 LUW2가 연관이 없어야 사용 가능.
  - Update work process의 return을 기다리지 않고 바로 다음 LUW가 실행됨.
  - 속도를 중요시할 때 사용.
- Synchronous update
  - LUW1과 LUW2가 연관이 있을 때 사용.
  - 코드 상: LUW1 뒤에 'COMMIT WORK **AND WAIT**'
  - 단점: 속도가 느리다.
  - 장점: DATA의 정확성은 올라감.
- Local update
  - Dialog work process하나에서 모두 처리. (Cf. Update work process는 시스템당 몇 개만 존재. 여럿이 공유)
  - **Update work process로 일을 전달하지 않음.**
  - 코드 상: 'SET UPDATE TASK LOCAL.~*LUW1*~COMMITWORK. ~ (*processing LUW1*)~LUW2
  - Synchronous와 유사하게 실행이 됨.
  - **대용량 batch 작업시 주로 사용.**

V1 and V2 updates
- 'Immediate start'가 들어가면 V1.
- Start delayed, collective run: V2
- V1은 VBLOG, main memory 두 군데에 request가 쌓임. V2는 VBLOG에만 쌓임.
- V1/V2는 CFIUT에서만 사용 가능. POC에서는 불가능.
- **Update work process가 V1/V2 각각 따로 존재함.**
- V1: Asnyc/sync/local 모두 사용가능
- V2: Async만 사용 가능. 보통 V2는 대용량.
- V2의 실행 조건: **V2는 V1이 성공했을 때에만 실행된다.**
- V1이 끝나면 2가지 발생: lock해제, V2 실행. 즉, V1이 끝나면 모든 lock이 사라지게 됨.

Performance rules
- Create new entries first (INSERT) ; 만들어야할 데이터를 만들어라
- Then perform changes to non-performance-critical tables (UPDATE) : 퍼포먼스에 영향 없는 변경분부터 수행하라
- Then perform changes to performance-critical tables(UPDATE): 퍼포먼스에 영향을 주는 변경분 수행.

Unit 5. Complex LUW Processing

Programs called within programs
- Synchronous calls
  - 실행했다 호출한 프로그램으로 돌아오는 경우.
    - CALL FUNCTION <function>
    - SUBMIT <program> AND RETURN.
    - CALL TRANSACTION <t_code>
  - 호출한 프로그램으로 돌아오지 않는 경우
    - SUBMIT <program>
    - LEAVE TO TRANSACTION <t_code>
- Asynchronous call of a function module
  - 코드: CALL FUNCTION 'ABC'
    STARTING NEW TASK <task_name>
    EXPORTING…
  - **새로운 user SESSION**을 하나 열어 parallel하게 processing함. 호출한 프로그램(위 코드의 'ABC')이 대용량일 때 주로 사용. 속도가 좋음. 해당 작업이 서로 연관이 없어야 함.

Accessing ABAP and SAP Memory
- ABAP Memory: 하나의 external session 안의 internal session 들끼리 공유하는 메모리 (importing/exporting)
- SAP 메모리: 하나의 user session 안의 모든 external/internal session들끼리 공유하는 메모리. (SET/GET parameters)
- 프로그램에서 'CALL FUNCTION'하면 같은 internal session 안에 올라오게 된다.
- SUBMIT(프로그램 호출) AND RETURN, CALL TRANSACTION시에는 또 다른 internal session 이 생긴다. 작업 수행 후, 호출한 프로그램으로 돌아간다.

Passing Data Between Programs
- Data transfer through the call interface
  - Function modult 호출
    CALL FUNCTION 'ABC'
    EXPORTING…
    IMPORTING …
    CHANGING…
    TABLE…
  - 다른 프로그램 호출(selection screen 가진)
    SUBMIT B [AND RETURN]
    WITH …
- SUBMIT…WITH: 이를 통해 program의 selection screen의 parameter 변수들 또는 select options 변수들의 모든 data를 넘길 수 있다.
  - SUBMIT <program> [AND RETURN] [VIA SELECTION-SCREEN]
    WITH <parameter> EQ <value>.
    WITH <sel-opt> <operator> <value> SIGN <s>.
    WITH <sel-opt> BETWEEN <value1> AND <value2> SIGN <s>.

WITH <sel-opt> NOT BETWEEN <value1> AND <value2> SIGN <s>.
WITH <sel-opt> IN <sel_table>.

- Passing Data using the ABAP memory: 미리 ABAP 메모리에 ABC라는 Data cluster에 데이터를 저장해놓고 프로그램에서 호출해 쓴다. SAP 스탠다드에서 주로 사용.
  - 프로그램 코드
    - EXPORT
      carrid FROM mycarrid
      wa_spfli
      it_spfli
      TO MEMORY ID 'ABC'.
    - IMPORT
      carrid TO p_carrid
      it_spfli
      FROM MEMORY ID 'ABC:.

- Data transfer through SAP memory
  - 같은 user session 안이라는 전제 하에 사용.
  - SAP 메모리는 screen 상에서도 identical한 ID(ex, 'CAR')를 이용해서 사용 가능.
    - 단, 스크린 페인터의 입력필드 속성창에 SET/GET parameter 체크해야함.
  - SET PARAMETER ID 'CAR' FIELD 'LH'.
    GET PARAMETER ID 'CAR' FIELD carrid.

- **'SAPL~'** : function group program이라는 뜻.

LUW logic in program-controlled calls
- COMMIT WORK하면 현재 LUW의 request들만 수행된다.
- CALL TRANSACTION··· UPDATE **'S'**. : SYNCHRONOUS. UPDATE될 때까지 기다림.
- CALL TRANSACTION··· UPDATE **'A'**. : ASYNCHRONOUS. UPDATE될 때까지 안 기다림.
- Accumulate locks for program call: 한 internal session에서는 E lock 축적사용 가능. 다른 internal session의 프로그램, 다른 external session 등에서는 사용 불가
- **Function module은 호출한 프로그램과 같은 internal session! -> lock 공유 가능!**

Unit 6. Appendix (용어만 좀 알아두자)
- Number assignment
  - Number range: 순차적으로 증가하는 값을 만들 수 있다. (ex. 게시물 번호)
  - Number range object를 먼저 만들어야한다. T-code: **SNRO**
  - Number range intervals: interval은 client별로 독립적이다. (= 별도로 만들어야 한다.)
  - CALL FUNCTION '**NUMBER_GET_NEXT**' : 겹치지 않는 number를 갖고 올 수 있게 해주는 standard function임.
  - CALL FUNCTION '**NUMBER_CHECK**' : 현재 number가 유효한지, 남아있는지 체크 가능.
  - CALL FUNCTION '**NUMBER_INFO**' : 현재 number range의 info를 보고 싶을 때
  - **NRIV** : DB에 있는 table buffer. Number range에 관련된 standard table임. 병렬처리가 가능. 여러 서버에서 동시에 사용해도 번호가 겹치지 않도록 해줌.

- Authorization checks
  - Authorization object: 권한들이 정의되어 있다.
  - Authorization object들을 묶어 authorization profile을 만들어 user에게 할당. (모듈 컨설턴트의 일)
  - Authorization check하는 코드

  AUTHORITY CHECK
  OBJECT 'S_CARRID'
  ID 'CARRID FIELD p_carrid
  ID 'ACTVT' FIELD '02'.

- SAP buffers
  - **Application server마다 table buffer 하나씩 있다.**
  - 각각 AS에서 DB의 내용을 buffering해서 쓰다보니 data의 mismatching이 발생할 수 있다.
  - 이를 피하기 위한 buffering type들
    - Resident buffering (100%) : full buffering
    - Generic buffering: 1 key field
    - Generic buffering: 2 key field
    - Single-record buffering (한 라인씩만 buffering)

- Native SQL
  - Buffering 사용 불가
  - DB system에 의존적이다.
  - 문법
    - EXEC SQL.
      <native SQL statement>
      ENDEXEC.

- Cluster tables
  - Data cluster에 저장 가능한 data들
    - Fields
    - Structure fields
    - Internal tables
  - Cluster table에 담아서 database에 import/export 가능.
    - 문법
      - EXPORT <name> FROM <obj> TO DATABASE <dbtab>(<ar>) ID <id>.
      - IMPORT <name> TO <obj> FROM DATABASE <dbtab><ar> ID <id>.

  Cf. 일반 table은 field만 저장 가능.

  - ABAP cluster databases: SAP 스탠다드에서 주로 사용됨. 일반 사용자는 별로 사용안함.

- Cluster table과 Transparent table의 차이점.

Cluster table	Transparent table
- Cluster의 data만을 포함 - **Less access** for large data quantities. - **Heterogeneous(여러 종류의) data**(field, structured field, internal table) - Flexible techniques - No linking of data possible - Access required cluster ID and area - Access only returns one cluster	In relational database - **Many accesses** since data is stored in various tables - Data can be linked and evaluated(평가하다) - Select with any logical condition

- SAP locks
  - Scope = 1일 때: lock을 잡은 그 dialog program에 종속적이다. Asynchronous update에는 적합하지 않다.
  - Scope = 2일 때: lock이 V1까지 transfer된다.
  - Scope = 3일 때: lock이 V1까지 transfer되며, dialog program 또는 update program에서 lock을 release할 수 있다.

- BAPI: Business Application Programming Interface. Function group이나 class와 유사한 개념. Business에 관련된 program들을 모아놓은 것.
  - BOR(Business object repository)들의 method들로 구현되어 있다.

< 실습 메모 시작>
LDB 사용을 위해서는 처음에 메뉴의 Goto-> Attributes 선택해서 사용할 Logical database를 지정해 줘야 함. (가장 중요).

Select * into table it_spfli
From spfli FOR ALL ENTRIES IN it_list "이 internal table 안에 있는 레코드들을 where절에서 조건으로 사용할 수 있다. (한 건이 아니라 여러 건)
Where carrid = it_list-carrid.

'Se93': t-code만드는 화면.. call transaction = submit and return via selection-screen과 같은 결과.
Call transaction <t-code> and skip first screen = submit <function이름>과 동일한 결과.

ABAP 메모리 사용하기: 한 internal session 안에서 메모리 공유. Export/import 문 사용.
SAP 메모리 사용하기: 한 user session 안의 external/internal session들끼리 메모리 공유. Set/get parameters 사용.

ALV list 그릴 때. Screen painter에 네모에 'c' 붙어있는 아이콘(custom control)로 리스트 그릴 부분(container) 그려줌.
미리 정의된 Class를 사용: cl_gui_custom_container, cl_gui_alv_grid.
Alv를 사용할 때, set_table_for_first_display(alv 리스트 출력시 부르는 메소드)에는 structure이름(ex. 'SPFLI')이나 field catalog 둘 중에 하나는 반드시 필요하다.

Lock object: Se11. lock걸면 ENQEUE/DEQEUE function이 생김(function builder: se37)
Sm12: 어떤 lock(현재 사용자에게)이 걸려있는지 알 수 있다.
한꺼번에 여러 object에 lock을 걸고 싶을 때는 '_collect'라는 기능을 체크하고('X'), FLUSH_ENQEUE라는 function을 호출해줘야한다.
'DEQEUE ALL'이라는 function은 내가 걸어놓은 lock들을 한꺼번에 풀어준다.

Enhancement 프로젝트: 'cmod'에서 수행.

WebDyn pro component의 구성 크게 3가지 - component controller, window, view
실습
1) webdynpro component create
2) view만들기
3)만든 view 윈도우에 끌어다놓기. 간단한 화면 그리기 (필요하면 component controller영역의 context에 전체 app에서 사용할 데이터를 모아 node/attribute 만들기)
4) inbound plug, outbound plug 만들기 (각 view에서 사용할 context의 노드를 component controller의 context로부터 끌어다놓기. 그리고 화면 설계, 바인딩.)
5)윈도우에 가서 outbound plug 아이콘을 inbound plug 아이콘에 끌어다놔서 navigation link 만들기.
6) 버튼 등을 만들어서 Action을 만들어 'onAction' 이벤트 생성하고 호출할 outbound plug를 지정해준다. (필요하면 만든 Action에 코드 추가 가능. Ex)select 문)
7) 컨텍스트 메뉴를 통해 create-> webdynpro application 확인.
8) 만들어진 webdynpro application의 컨텍스트 메뉴의 test 선택하면 실행해볼 수 있다.
** wizard 기능을 통해 table등을 그릴 수 있다. Context를 선택해서 자동으로 그려질 수 있게 만들 수 있다.

Node를 가리키기 위해서는 if_wd_context_node의 레퍼런스 변수를 선언해야.
Ex) node_selection TYPE REF TO if_wd_context_node.
Node_selection = wd_context->get_child_node('SELECTION').

< 실습 메모 끝~>

<4주차 [TAW12_1]>
OOP!

Unit 1. Intro to OOP
- Typical procedual ABAP program.
  - Type definitions
  - Data declarations
  - Main program
    - Calling subroutines (PERFORM form1…)
    - Calling function modules(CALL FUNCTION 'FB1'…)

- Definition of subroutines
- OOP 기본 문법

```
REPORT…
DATA: counter TYPE I,
      wa TYPE kna1.
…
CLASS lcl_car DEFINITION
…
ENDCLASS

*---------main program ------------
Counter = counter + 1.
CREATE OBJECT …
MOVE wa TO …
```

- 객체들끼리 client-server관계가 형성 가능.
- 다른 OOP속성들
  - Inheritance
  - Polymorphism for support of generic programming: 하나의 이름으로 다양하게 정의하여 사용 가능
  - Event controlling: client가 이벤트를 날리면 해당하는 server만 반응
- Defining classes (local class)
  - 선언부

```
CLASS lcl_vehicale DEFINITION.

ENDCLASS.
```

  - 구현부

```
CLASS lcl_vehicale IMPLEMENTATION.

ENDCLASS.
```

- Visibility sections of attributes
  - Public attributes
    - Generally visible
    - Generally changeable - EXCEPTIONL RED-ONLY addition
  - Private attributes
    - Only visible within the class
    - Only changeable within the class
- Static attributes
  - Exit per class. 접근 operator는 '=>'
  - Definition with 'CLASS-DATA'.
- **SE38** : class 만들려면. Class builder
- METHODS
  - returning 제외하고는 function과 매우 유사.
  - method에는 function에는 있는 **tables가 없다.**
  - **Returning parameter가 있으면 exporting/changing은 사용할 수 없다.** (당연~)
  - 코드: Class의 definition 안에서

```
METHODS method_name
        [IMPORTING…
        EXPORTING…
CHANGING…
RETURNING … "요게 있으면 'FUNCTIONAL METHOD'라 함.
EXCEPTIONS… RAISING exception class].
```

  - Function의 실제 코드는 class의 구현부에…
- Static method는 'CLASS-METHODS'로 정의
- Reference 변수 만들고 'CREATE OBJECT <reference변수이름>'해서 코딩.
- Garbage collector: 가리키는 객체변수가 없는 object는 시스템이 자동으로 파악해서 처리함.
- Multiple instantiation: multiple reference-> garbage collector에 의해 object가 날라가지 않도록 internal table 이용.

Ex)

```
itab TYPE REF OF REF TO lcl_vehicle.
CREATE OBJECT …
APPEND … TO itab.
…
```

- Functional Methods
  - Definition
    - "RETURNING" parameter가 있으면 functional method.
    - EXPORTING/CHANING parameter 사용 불가. 다른 건 ok.
  - Call
    - 함수처럼 return값을 대입해서 사용 가능 (explicit assignment of RECEIVING parameter)
    - Implicit calls possible in various expressions
      - MOVE, CASE, LOOP statement
      - Logical expressions(IF, WHILE, CHECK, WAIT UNTIL)
      - Arithmetic and bit expressions(COMPUTE)
- '->' 뒤에 괄호가 있으면 method, 아니면 attribute!
- Constructor: 특수한 method. Importing과 exception만 가능. 무조건 public section에 선언되어야 함. (method 이름이 constructor)
- Static constructor: 클래스당 단 한번만 실행됨. (코드 상 메소드 이름: 'class_constuctor'). Parameter나 exception을 가지지 않음.
- Self reference: 'me->'! 안 써주면 자기 자신임. Local과 구분하기 위해 'me' 사용. 'me' 생략해서도 사용 가능.

Unit 2. Object-oriented concepts and programming techniques

- SAP에서 sub-class의 부모는 오직 하나임. (다중상속 안됨)
- Inheritance syntax

```
CLASS lcl_truck DEFINITION INHERITING FROM lcl_vehicle.
…
ENDCLASS.
```

- Redefining methods: 이름은 같으나 sub-class에서 재정의해서 쓰는 경우. Parameter는 바꿀 수 없고, 구현 부분만 변경 가능.
  METHODS estimate_fuel **REDEFINITION**. "sub-class의 class definition 부분에 이렇게 정의
- Constructor는 redefinition과 달리 parameter를 바꿀 수 있다.
- Subclass의 constructor구현시 반드시 superclass의 constructor를 호출해 줘야 한다.
  : CALL METHOD super->constructor(…).

- Constructor 호출 방법
  - 해당 클래스가 constructor 갖고 있으면 부르면 됨
  - 안 갖고 있으면 부모 class의 constructor를 부르면 됨. (부모 클래스 constructor의 parameter를 넘겨서 호출 가능.)
- Static components: 상속은 받을 수 있으나, redefine은 할 수 없다.
- Class의 public, protected section의 내용은 subclass와 공유됨.(상속됨)
- Polymorphism의 전제 조건
  - Inheritance(상속)
  - Redefinition(Method 재정의)
  - Upcast(부모변수=자녀객체주소값)
    - Super 객체를 가리킬 수 있는 객체 변수가 sub객체를 가리키는 것. = up-casting = widening casting
- Down cast
  - Up cast된 super 객체를 통해서만 사용 가능하다.
  - super객체를 가리키는 super 객체변수는 down casting을 할 수 없다. (dump발생함.)
  - 문법: sub객체 **?=** upcasting된super객체

Interfaces and casting

- Interface
  - 상속관계가 없는 class들에 component를 공유하게 해 줌.
  - Interface는 여러 개를 상속받는 것이 가능하다. (멀티상속 가능)
  - Definition만 있다. **구현부는 상속받은 클래스에 있다.**
  - **오직 public component만 존재.** (그러니까 public section등 써줄 필요 없다.)
  - **Naming rule: 'lif_xxxx'** ex. lif_partner(local), zif_XXX/yif_xxx(global)
  - 코드

```
INTERFACE lif_partner.
        METHODS display_partner.
ENDINTERFACE.

...
CLASS lcl_rental DEFINITION
        PUBLIC SECTION.
                INTERFACES lif_partner.
ENDCLASS.

...
CLASS lcl_rental IMPLEMENTATION.
        METHOD lif_partner~display_partner.
        ...
        ENDMETHOD.

...
ENDCLASS.
```

i

  - Interface는 multi상속이 가능한 super class로 생각할 수도 있다. (up-casting/down-casting에 대해)
  - Compound interface(중복된 interface): interface가 interface를 상속받을 수도 있다.

```
INTERFACE lif_lodging.
        INTERFACE lif_partner.
        METHODS book_room.
ENDINTERFACE.
```

Events

- 기본 개념: Sender object가 event를 보내면, Recipient, 즉 'handler'가 해당 이벤트를 처리한다.
- 실행 순서
  - Class defines an event: 이벤트 정의 (EVENTS, CLASS-EVENTS)
  - Object or class triggers the event: 이벤트를 발생시키기 (RAISE EVENT)
  - Handler class defineds and implements the handler method: 메소드의 기능 구현.
    ([CLASS-]METHODS…FOR EVENT…(이벤트) OF…(누가 발생시키나))
  - "Handler object" or handler class is registered to events at run time(SET HANDLER): 실제 발생한 이벤트를 사용하겠다고 등록. (설정 안하면 무시됨)
    - SET HANDLER ref_handler->on_eventname
      [FOR ref_sender | FOR ALL INSTANCES]
      [ACTIVATION flag].

Unit 3. Object-oriented Repository objects
: DB영역에 존재. SAP의 client에 의존적이지 않음 = 모든 Client에서 사용가능.

Global classes and Interfaces

- Class builder(SE24)로 global class(zcl~/ycl~)/global interfaces(zif~/yif~)를 만들어보자.
- Global class만들기
  - Se24 (class builder)
  - Se80(context메뉴에서 class create)
  - Method에 커서 놓고 오른쪽 위에 'signature' 버튼 누르면 해당 method의 parameter들을 볼 수 있다.
- Class builder에서 ㄱ자모양 자 버튼을 클릭하면 객체를 만드는 것을 테스트 가능. 실행해볼 수 있다.
- Local class를 global class로 import도 가능하다.

Special object-oriented programming (출제 예상)

- Abstract class: 구현하지 않은 method가 class에 존재 가능. 그런 method가 있으면 abstract class. 상속받은 sub class에서는 구현해야 함.

```
CLASS lcl_...DEFINITION ABSTRACT.
...
METHODS … ABSTRACT…
...
ENDCLASS.
```

- Final class: 더 이상 상속하지 않겠다. = 더 이상 super class가 되지 않겠다.

```
CLASS lcl_... DEFINITION FINAL
        [INHERITING FROM…].
```

…
ENDCLASS.

- Final method: 더 이상 redefinition 불가
  CLASS lcl_… DEFINITION.
  …
  METHODS … **FINAL** …
  …
  ENDCLASS.
- "singletone pattern": 프로그램에서 딱 하나의 객체만 만들게 하겠다. 즉, INTERNAL SESSION에서 해당 object는 딱 하나만 만들어진다.
  CLASS lcl_… DEFINITION **CREATE PRIVATE**.
  …
  ENDCLASS

- Friend: friend 뒤에 온 녀석은 친구. private영역도 접근하도록 허락해줌.
  CLASS lcl_1 DEFINITION CREATE PRIVATE
  FRIENDS lcl_2.

Unit 4. Class-based Exception concept

- 6.10 버전 이후 class-based exception 처리가 가능. : CX_EXCEPTION.
- Naming rule: exception class는 항상 'CX_'로 시작한다. (개발자가 만든 건 'ZCX_'/'ZCY_'.
- Exception이 발생하는 경우 -> exception 처리를 하기 위한 exception class의 instance가 생성됨. -> predefined exception class나 개발자가 만든 exception class에 의해 해당 exception에 맞는 처리.
  - Explicit: RAISE EXCPTION TYPE…
  - Runtime 시에 발생하는 runtime error
- **CX_ROOT**: exception class의 최상위 클래스. 모든 에러/exception은 여기에 걸린다.
  - CX_ROOT의 subclass들. (3개가 다. 모든 exception class는 이들의 sub-class이다.)
    - **CX_NO_CHECK**
    - **CX_DYNAMIC_CHECK**
    - **CX_STATIC_CHECK**
  - CX_ROOT를 상속받을 수 있는 exception class는 단 3개뿐.
  - 모든 exception class는 3개의 exception class 중 하나를 상속받아야함.
- IF_MESSAGE: exception 발생시 출력할 메시지를 담은 interface.

- SE24: global class로 exception class를 만들기 위해 class builder를 사용. 상속받은 속성은 파란색.

- 코드상으로 TRY~ENDTRY를 써서 exception 발생시 catch함.

Unit 5. Shared object
: 모든 user session들이 Shared object를 공유할 수 있다. (table buffer처럼). Shared memory는 Application server에 있다.

- SHMA: area, Shared object를 만들고, 볼 수 있는 t-code.
  - Area 안에 객체 2개 반드시 필요. (root class/ catalog class)
    - Root class는 카탈로그 객체 변수를 가지고 있으면서 catalog class의 object를 가리킨다.
    - Catalog class의 객체는 actual data를 가지고 있다.
  - User session에서는 handle을 만들어 handle 안의 객체변수가 shared memory 안의 root object를 가리키게 해서 사용한다.
    - Handle: 영역이 다른 (internal session과 shared object memory) 객체를 연결시켜주는 object가 handle이다.
  - Read lock/write lock등을 걸어서 쓴다.
    - 방식: cl_my_area=>attach_for_write()
      Attach_for_read()
      Detach_commit() "모든 작업을 commit하고 lock도 풀어줌.
      Detach() "lock 해제
  - Read lock이 걸려 있는 상태에서 catalog에 update를 하고 싶으면?
    - 다른 object를 만들어서 write lock걸고 사용한다.
  - Shared object에서 쓰는 lock은 shared lock의 개념.
  - 각자 lock을 잡고 쓰면서 shared object의 lock을 동시에 잡고 쓰기 때문에 사용자가 읽는 데이터가 mismatch될 수도 있다.

- Shared memory 사용을 위해서는 최소 3개의 object를 만들어야함.
  - Area class
  - Root class
  - Catalog class
- Class 만들 때 -> class properties tab에서 'Shared memory enabled'에 체크하기.

- 언제 쓰느냐?
  - Cross-program buffering of data that is often read, but rarely written. : 자주 읽지만, 거의 write할 일이 없는 데이터에 쓴다.
  - Simultaneous read accesses are supported: 동시에 읽기접근이 가능
  - Access is regulated by a lock mechanism. : lock 방식에 의한 접근이 통제됨
  - Data is stored as object attributes. : data는 object의 attribute로 저장됨
  - Memory bottlenecks result in runtime errors and have to be caught. : 런타임시의 메모리 bottleneck은 에러를 발생시키므로 잡아줘야 한다.

Unit 6. Dynamic programming
: field symbols(포인터와 유사), data references.

- Field symbols
  - 문법
    FIELD-SYMBOLS <fs> {{TYPE|LIKE}…|TYPE ANY}.
    ASSIGN dataobject TO <fs>.
    UNASSIGN <fs>.
    … <fs> IS ASSIGNED… "현재 가리키는 곳이 있는지 알아보는 방법.

  - 예제 코드 : **field symbol 사용시에는 꺽쇠를 반드시 쓴다!**

```
DATA int TYPE I VALUE 15.
FIELD-SYMBOLS <fs_int> TYPE I.

ASSGIN int TO <fs_int>.
WRITE:/ int, <fs_int>. " 출력: 15 15

<fs_int> = 17.
```

```
WRITE:/ int, <fs_int>. "출력: 17 17

UNASSIGN <fs_int>.
IF <fs_int> IS ASSIGNED.
        WRITE:/ int, <fs_int>.
ELSE.
        WRITE:/ 'filedsymbol not assigned' (fna).  "이게 출력됨.
ENDIF.
```

- 문법

```
ASSIGN dataobject TO <fs> CASTING [TYPE type_name | …].
```

- 예제코드

```
TYPES: BEGIN OF st_date,
        year(4) TYPE n,
        month(2) TYPE n,
        day(2) TYPE n,
END OF st_date.

*option 1: implicit (casting시 type 명시 안 한 경우)
FIELD-SYMBOLS <fs> TYPE st_date.
ASSIGN sy-datum TO <fs> CASTING.
WRITE:/<fs>-year, <fs>-month, <fs>-day.

*option2: explicit
FIELD-SYMBOLS: <fs> TYPE ANY,
                <fs_year> TYPE st_date-year…

ASSIGN sy-datum TO <fs> CASTING TYPE st_date.
ASSIGN COMPONENT 1 OF STRUCTURE TO <fs_year.
ASSIGN COMPONENT 2 OF …
WRITE:/ <fs_year>,…
```

- Data references :field symbol과 똑 같은 기능.
  - 문법

```
TYPES reftype       { TYPE REF TO type_name |
                    LIKE REF TO  do_name |
                    TYPE REF TO data }.
DATA ref            { TYPE REF TO type_name |
                    LIKE REF TO do_name |
                    TYPE REF TO data        }.

GET REFERENCE OF dataobject INTO ref.       "ref가 data object를 가리킴.
```

  - 예제 코드

```
TYPES ref_int_type TYPE REF TO i.
DATA ref_int        TYPE ref_int_type.
DATA int            TYPE I VALUE 15.
…
GET REFERENCE OF int INTO ref_int.          "obtain a reference to a data object.
WRITE ref_int->*. "결과: 15

MOVE 17 to ref_int->*.
WRITE ref_int->*. "결과: 17.
```

  - Validity of references : data reference가 가리키는 게 있는지 없는지 체크.
    - 요렇게 쓴다: … ref **IS [NOT] BOUND**… cf) fs는 'IS ASSIGNED'였음.

- Data **Object**의 Daynamic instantiation and cast assginemnts
  - 문법

```
DATA ref TYHPE REF TO typename.
CREATE DATA ref.   "data 객체를 하나 만듬.
```

- Generic typing of Data objects at runtime
  - 문법

```
DATA ref TYPE REF TO data.      "아무 datatype이나 다 가리킬 수 있다.
CREATE DATA ref TYPE typename.          "type을 create data할 때 만들겠다는 뜻.
```

- Dynamic Instantiation of Data Objects at runtime
  - 문법

```
DATA ref TYPE REF TO data.
DATA var_type TYPE … .

var_type=…
CREATE DATA ref TYPE (var_type)             "아래 실제 예제 코드 참고.
```

  - 예제

```
var_type = 'I'.
CREATE DATA ref TYPE (var_type).            "ref는 int type의 데이터를 가리키게 된다.
```

- Dereferencing Generically typed Data references
  - ASSIGN ref->* TO <fs> [CASTING…]. "data reference와 field symbol이 가리키는 곳이 같도록 만드는 문장.
  - 예제코드

```
DATA ref_itab TYPE REF TO data.

FIELD-SYMBOLS <fs_itab> TYPE ANY TABLE.

CREATE DATA ref_itab TYPE …(table 이름) TABLES OF line_type
                WITH DEFAULT KEY.

ASSIGN ref_itab->* TO <fs_itab>.
```

SELECT * FROM tab_name INTO TABLE <fs_itab>.

- reference와 field symbol을 통해 하나의 data object를 갖고 동적으로 만들면서 관리가 가능하다.

Runtime Type services

Runtime type Identification (RTTI) : class-based concept.

- 'CL_ABAP_'으로 시작하는 class들이 관련 class들임.
- 최상위 클래스인 CL_ABAP_TYPEDESCR 밑으로 subclass들이 존재.
- RTTS: Runtime type service
- 코드 예

```
...
DATA go_descr TYPE REF TO cl_abal_classdescr.
go_descr ?= cl_abap_typescr=>describe_by_object_ref( go_vehicle ).
IF go_descr->get_relative_name() = 'LCL_TRUCK'.
        go_truck ?= go_vehicle.      "typesafe downcast
ENDIF.
```

- RTTC: Runtime type creation – runtime에 object create도 가능. object create를 위한 static method를 호출해서.
  - 예: cl_abap_xxxdescr=>create(…)
  - Each type has a runtime type object(RTTS instance)
  - the runtime type object fully describes the data type.
  - a type object is local to the program, transient(일시적인), and anonymous(nameless)
  - **A type object cannot be deleted or changed** : runtime에 생성된 object는 삭제나 변경이 불가능!

<TAW 12_2: 5주차>

핵심 개념들: Enhancement(Using Customer exits)/Modification/new enhancement/BAdI/Web DynPro

- Modification: SAP에서 Enhancement를 제공하지 않을 때, Standard program을 강제적으로 수정하는 것.
  SAP에서 권장하지 않으며 현재는 거의 쓰지 않음. (new enhancement에 의해 거의 커버 가능)
  SAP 시스템의 업그레이드에 문제가 생길 수 있음.
  - User Exit
- Enhancement : SAP application에 고객의 변경을 예상하고 미리 구멍을 만들어 놓아, 그 구멍에 원하는 부분을 구현하여 끼워넣는 것.
- Enhancement 가 가능한 것
  - ABAP Dictionary상의 Tables, Data elements
  - Customer exits
  - Business Transaction Events (BTE)
  - Business Add-Ins(BAdI)

Unit 1. Changing the SAP standard system. (overview)

- ABAP workbench 입장에서 SAP R/3 시스템을 change할 수 있는 3가지 방법
  - Modification
  - Enhancement
  - Customer development(개발)
- Enhancement를 통한 table 변경(추가) 방법 2가지
  : customer request에 의해서 추가하고 싶은 column이 있을 때, 추가 가능. column이름을 반드시 'YY'나 'ZZ'으로 시작해야 한다.
  (이유: SAP 시스템 업그레이드시 업버전에서 추가된 필드명과 사용자가 추가한 필드명이 충돌하지 않게 하기 위해서)
  - **Append structure**: 추가하고 싶은 필드가 생겼을 때
  - Customizing includes("**CI includes**") : SAP가 이미 사용자의 변경(확장 요청)을 예상하고 테이블을 만들어놓은 것.
- A를 통해 B를 확장.
  - user exit: subroutine
  - customer exit: function module
  - Business Transaction event(BTE): function module
  - Business Add-In: Method

Unit 2.Enhancing Dictionary Elements

Table Enhancements (Enhancement 중 append structure/customizing includes(=CI include))

- **Append structure**: SAP에서 미리 예상하지 않은 것. 해당 table만 변경됨.
- **Customizing include** : SAP에서 미리 예상한 것. 해당 테이블 변경 시 이를 참조하는 다른 모든 테이블들도 변경분이 반영됨. SAP가 그렇게 구성해 놓았기 때문.) 즉 CI_ABC라는 CI include를 사용하는 모든 테이블은 사용자의 변경확장분이 적용된다.

- SE11 화면 상단 오른쪽에 버튼 눌러 Standard table 확장 가능. YY, ZZ 명명규칙 지켜서 필드 이름 지정.
- 새로운 version에서 필드가 추가되면 Enhancement에 의해 확장된 table의 (YY, ZZ) 뒤쪽에 새로운 칼럼이 추가됨.

Text Enhancements

- **T-code: CMOD 사용해서** 특정 data element를 enhancement할 수 있다. (overwriting)
- SAP 시스템 업그레이드 시 SAP 스탠다드의 내용으로 restore, 즉 복원됨. 사용자는 자기가 변경한 내용을 유지하고 싶음.
  -> '**restoring SAP field labels**(기능 이름)'를 통해 원하는 버전으로 복원 가능.
- Restoring SAP field labels
  - SAP standard의 특정 시점으로 복원 가능
  - Customer가 변경한 특정 시점으로도 복원 가능.
- Customer가 변경한 data element에 대한 documentation을 standard documentation 밑에 추가할 수 있음. (역시 t-code CMOD, Goto->…->EN-New Customer Documentation 선택 시 해당 documentation에 추가 가능(수정&재작성하는 것이 아님!). Enhancement 통해서)

Unit 3. Enhancements Using Customer Exits

- Customer exit의 종류 (**MFC**로 외우자)
  - Program exit (**F**unction module을 확장)
    - Standard 프로그램 안에 '**CALL CUSTOMER**'가 있으면 확장 가능.

- Menu exit (메뉴 확장)
  - 키워드는 동일하게 'CALL CUSTOMER'
  - Menu exit에 해당하는 function 이름은 '+'로 시작함.
- Screen exit (화면 내용 확장)
  - SAP에서 미리 확장을 예상하고 subscreen area 공간을 확보해놓음.
  - program끼리의 data 전달은 importing/exporting parameter를 통해 이루어진다.
  - Standard 프로그램 안에 'CALL SUBSCREEN'이 있으면 확장 가능.

- 하나의 Enhancement는 반드시 하나의 Enhancement project에만 종속적이어야함.
- Enhancement project 만드는 t-code: CMOD
- SAP에서 제공해준 Enhancement 조회하는 t-code: SMOD

- Enhancement project의 이름은 Z또는 Y로 시작하지 않아도 된다.
- Enhancement project에는 여러 개의 Enhancement를 assign할 수 있다.
- Enhancement project 완료 시 request No. 이관(transport)을 통해 타시스템(운영서버 등)에도 적용 가능.

- Customer exit을 위한 function group의 이름에는 'X'를 넣는다. ex. SAPLXAAA

Unit 4. Business Add-Ins (classic BAdI)
: Class로 이루어져 있는, Enhancement를 할 수 있는 부분.

- 기존 Enhancement를 구현한 부분은 오직 한 번만 사용 가능했으나 BAdI를 통해 여러 프로젝트에서 해당 Enhancement를 사용 가능해짐.
- 사용 방법
  - BAdI의 reference를 얻는다. ('CL_EXITHANDLER' : 일종의 도우미 클래스. 코드에 이게 존재하면 BADI도 존재.)
  - 필요한 BAdI instance의 method를 호출한다.
- program에서 'cl_exithandler'를 찾아서 BAdI의 사용 여부를 확인 가능.
- SE18: BAdI에 대한 정의를 볼 수 있는 t-code.
- SE19: BAdI를 구현할 때 쓰는 t-code.

- BAdI 사용시 multiple use가 enable되어 있으면 어떤 BAdI가 먼저 호출될지 모르므로 시간종속적인 로직을 넣으면 안됨.

- filter types: 특정 필터 값에 따라 BAdI의 implementation이 틀리게 동작하고 싶을 때 사용.

- BAdI implementation class이름: ZCL_IM_ 또는 CL_IM_으로 시작.
- BAdI interface 이름: IF_EX_또는 ZIF_EX로 시작.

Unit 5. Modifications
:user exit

- corrections and repairs.
  - correction: original에 대한 수정을 하는 것. (original: 최초로 만들어진 것. original을 install하면 copy본이 내 시스템에 복사되는 것)
  - repair: copy본에 대한 수정을 하는 것.
- Modification은 SSCR에 등록하고 key값을 SAP로부터 받아야 할 수 있다.
- Modification 과정
  - standard program에서 수정버튼 누름(연필모양)
  - SSCR 키값 넣기
  - warning message 뜸
  - Change request no.입력.
- Modification 작업을 하는 동안 change lock이 걸려 다른 user들은 사용할 수 없게 된다. release되기까지 import 불가. 빨리 수정/test/release해야함.

- Modification assistant
  - 사용을 위한 전제조건 3가지
    - SSCR Key 존재
    - Change request no. 존재
    - "eu/controlled modification" profile이 on으로 켜있어야함. (주로 BC가 시스템에 설정)
- User exit : modification 중 하나의 기능. Enhancement인데 기술적으로는 modification에 속한다. include 프로그램 이름 뒤에서 두 번째에 'Z'가 있다. (ex. MV45AFZB)
  - include 프로그램 안의 subroutine들이 이미 들어가 있고 이를 수정하여 사용.
    - subroutine들의 이름은 'userexit_'으로 시작
  - 특징
    - blank subprograms
    - includes delivered once: upgrade시 다시 제공되지 않는다. 다른 이름으로 새로운 include는 제공 가능)
    - mainly used in SD
    - Technically: modification (해당 include가 MV로 시작하는 standard니까)
  - 코드 내에 'perform userexit'가 있으면 user exit이 존재함.
- Note assistant : SSCR key가 필요 없다.
  - 버그 수정, 작은 개선 작업 등에 사용.
  - Correction of individual errors. does not resupport packages
    : note를 통해서 system에 대량의 변경을 가하는 것은 아니다. 사소한 문제에 대한 변경을 하는 것.
  - 'SNOTE' : 현재 시스템에 어떤 notes가 적용되어 있는지, 각 note의 진행상황도 볼 수 있다.

- Modification Adjustment: 스탠다드와 사용자에 의해 개발된 또는 modify된 부분이 겹치면->adjustment 필요.
  - 이를 위한 t-code: SPDD(Dictionary object adjustment- Domains/Data elements/tables), SPAU(all other ABAP Repository objects)

Unit 6. Enhancements
: 7.0 new feature. 기존의 customer exit을 통해 했던 기능을 대체 가능. implicit enhancement/explicit enhancement, new BAdI, Switch Framework

- Enhancement points (2가지) : modification 없이 SAP program, SAP function module, SAP methods 등에 소스코드 추가, 변수나 파라미터 선언이 가능해졌다.
  - implicit: No SAP preparation. 새로 도입된 개념. 프로그램 단에 코드 추가하여 사용가능.
  - explicit: prepared by SAP developers.
    - ENHANCEMENT-POINT : 원하는 로직을 추가 가능.
    - ENHANCEMENT-SECTION : 원래 SAP 코드를 REPLACE하고 싶을 때 사용
- Enhancement spot: Explicit enhancement points(추가의 개념)/Explicit enhancement sections(변경, replace의 개념)/ New BAdI는 enhancement spot을 통해 관리된다. (composite enhancement spots: Enhancement spot들의 조합)
- New BAdI

- classic BAdI에 비해 performance가 개선됨.
- 추가적인 기능이 구현됨
  - Enhanced(확장된) filter concept.
  - option to inherit attributes from sample implementation classes
  - …
- Integration into the new Enhancement Framework
- Integration into the Switch Framework. (on/off로 사용여부 선택 가능)

- classic BAdI 구현:'**cl_exithandler**' 찾아보면 됨. class를 통해 handler 취득.
- New BAdI 구현: 'GET BADI'/'CALL BADI' 찾아보면 됨. (전자는 handle을 얻어오는 것, 후자는 BAdI handle의 method를 호출하는 것.

- New BAdI는 커널 안의 BAdI handle을 이용하여 method들을 호출 (관리가 집중화 되었다. 효율적인 관리가 가능.)
  - New BAdI 개념은 classic과 동일
  - Adapter class를 사용하는 것이 아니라 core kernel안에서 BAdI에 대한 handle을 얻어오고 이를 통해 각각의 implementation된 method들을 호출해줌.
  - New BAdI는 개선된 filter 기능 제공: filter – 필터값에 따라서 어떤 implementation을 호출할지를 결정할 수 있다.

- Switch Framework : New Enhancement 기술은 모두 여기서 통합 관리됨. **원하는 기능만 스위치를 켜고 꺼서 사용 가능. 관련 t-code: SFW5**
  - **주의:** New enhancement만 관리 가능. activate/deactivate와 다른 개념임.

- 실습!
  - call customer 찾기 -> customer exit 존재여부 확인
  - cl_exithandler 찾기 -> classic BAdI 있는지 확인
  - GET BADI 찾기 -> New BAdI 있는지 확인
  - 셋 다 없으면 SAP가 enhancement를 미리 마련하지 않은 것!
  - 코드를 추가해야…. explicit enhancement가 없으면 implicit enhancement로..

Unit 7. WebDynpro intro.

- declarative programming approach: 프로그램을 하나씩 개발하는 게 아니고, 정의해놓고 Drag&Drop으로 만듦.
- metadata approach: 프로그램에 대해 대략적으로 골격(?)을 정의부터 하고 구현함.
- 어디서 개발? SE80.
- 화면설계 하다보면 자동으로 metadata형태가 만들어짐.
- WebDynpro로 개발하면 device에 비종속적. (전화기, PDA, 랩탑…)
- MVC 모델 적용.
- WebDynpro의 기본 entities. 구성품(component)
  - windows : view들을 담고 있다.
  - views
  - component controllers : logic/flow에 대한 부분.
    - 이 안에 node와 attribute들이 존재.
    - node는 attribute들을 묶어주는 기능. node없이 attribute만 존재도 가능하다.
- WebDynpro에서의 'data binding': 화면상의 view element와 view controller의 context와 연결하는 것.
- view간 이동에 있어야할 3가지
  - inbound plug
  - outbound plug : outbound link는 반드시 navigation link를 가지고 있다. (inbound plug는 아님)
  - navigation link
- "**View Assembly**": 한 화면에 나올 수 있는 가짓수. 용어 알아두기.
- Custom controller: component controller가 너무 방대해지면 custom controller를 사용. 자주 사용하던 로직을 따로 구현 가능.

- 외부에서 webdynpro component에 접근하는 방법 2가지 (=component interface)
  - Interface view
  - Interface controller

Unit 8. WebDynpro components : skip!
Unit 9. The context at design time: skip!

Unit 10. Defining the user interface

- Container elements and layout managers
  - flowlayout : 화면이 허락하는 한도 내에서 한 줄로 표시됨. 충분한 화면 공간이 없으면 다음 라인에 표시
  - rowlayout : 다음 라인에 보일 element를 'rowheaddata'로 설정하여 줄바꿈을 지정해줄 수 있음. 충분한 화면 공간이 없으면 짤려보임. 스크롤바로 이동해서 볼 수 있다.
  - matrixlayout : 모든 화면의 elements가 정렬이 된다. 화면을 바둑판 모양으로 나누어서 사용.
  - gridlayout : 화면의 layout을 몇 컬럼으로 나눌지 미리 정의한 것.
- wdr_test_ui_elements: SAP의 스탠다드 Web Dynpro 예제 프로그램. UI의 속성들을 하나씩 테스트 가능.
- Web Dynpro의 method 종류 (2가지)
  - hook method: SAP WebDyn pro에서 제공
  - instance method: 개발자가 선언/작성하여 사용.

Unit 11. skip.