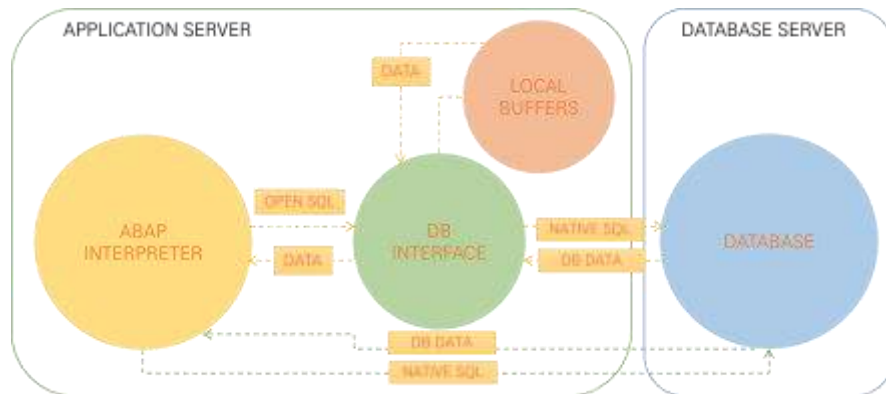


## SQL 정의 |

DML 데이터 처리 언어	DATA MANIPULATION LANGUAGE DB   테이블에 저장된 DATA를 검색, 삽입, 삭제, 갱신, 재구성하기 위해 사용되는 언어
DDL 데이터 정의 언어	DATA DEFINITION LANGUAGE 응용 프로그램과 DBMS에 데이터 요구를 표현할 수 있는 인터페이스를 기술하기 위한 언어로서 DB를 생성할 목적으로 사용하는 언어
DCL 데이터 제어 언어	DATA CONTROL LANGUAGE 무결성, 보안 및 권한 제어, 회복 등 DATA를 보호하고 관리하는 목적으로 사용하는 언어

## SQL 정의 |

SQL에는 OPEN SQL과 NATIVE SQL 이 있다. OPEN SQL은 ABAP 언어에서 만 사용되며 DATABASE INTERFACE를 통해 NATIVE SQL로 번역된다. NATIVE SQL은 데이터 베이스에 사용되는 SQL이다.



## OPEN SQL VS NATIVE SQL |

## OPEN SQL

OPEN SQL은 DATABASE DATA를 조작할 수 있는 ABAP 명령어로 구성되어 있으며, 서로 다른 DBMS | DATABASE MANAGEMENT SYSTEM 환경\* 에서도 같은 명령어를 사용한다. OPEN SQL은 DDL, DCL을 사용할 수 없으며 SELECT문과 같은 DML만 사용할 수 있다. LOCAL BUFFER을 사용할 수 있으며, NATIVE SQL보다 사용방법이 간단하며 ABAP 프로그램 작성 시 자동으로 SYNTAX CHECK가 수행된다는 특징을 가진다.

## \*ORACLE, MYSQL과 같은 DATABASE

\*\*SAP는 ORACLE DATABASE를 많이 사용한다. OPEN SQL만으로 복잡한 SQL을 모두 구현할 수 없을 때 DATABASE NATIVE SQL을 사용하여 정밀한 DATA SELECT를 수행할 수 있다. NATIVE SQL은 DB INTERFACE를 통하지 않고 바로 DB SERVER로 전송된다.

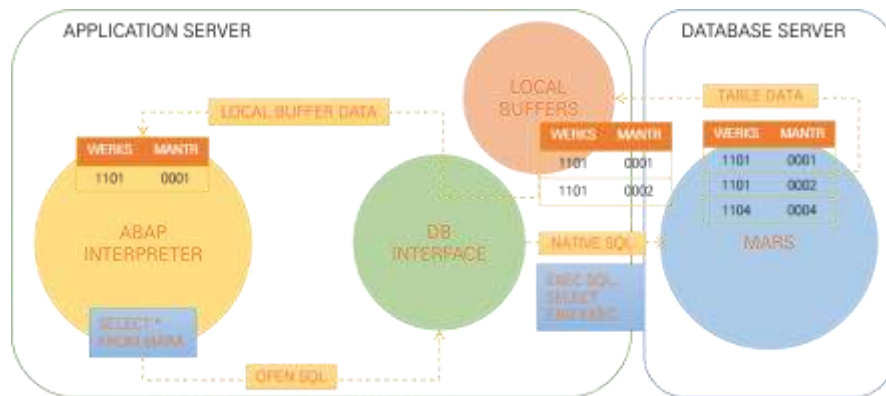
## NATIVE SQL

DATABASE에 직접 접속하여 DML, DDL 언어를 사용할 수 있다. DDL 은 CREATE, MODIFY할 수 있다. OPEN SQL의 COMMAND SET | SELECT UPDATE DELETE 등을 사용할 수 있으며 OPEN SQL로 해결되지 않는 복잡한 SQL은 NATIVE SQL을 이용할 수 있다.

## SQL 과 LOCAL BUFFER |

SAP LOCAL BUFFER는 R/3 ARCHITECTURE에서 지원하는 기술로 DB의 부하를 줄이는 중요한 역할을 담당하며 TABLE의 TECHNICAL SETTING에서 BUFFER 사용 설정을 한다.

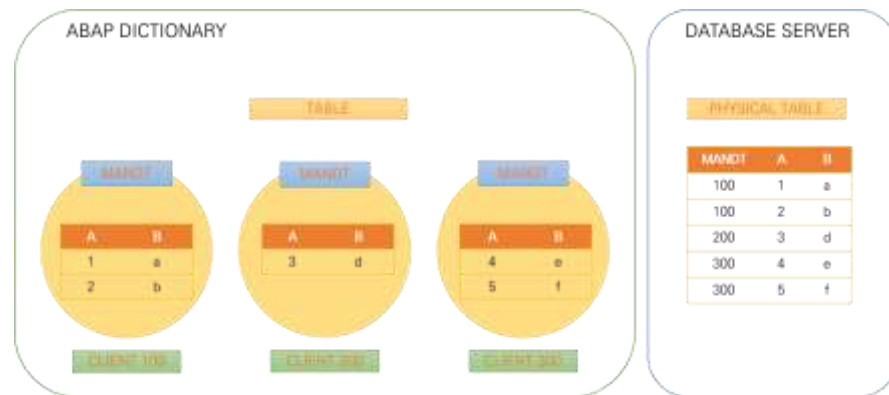
## SQL 실행 순서 |



- 1 | MARC TABLE에서 WERKS = '1101'인 DATA를 조회하는 SQL을 실행
- 2 | DB INTERFACE에서 NATIVE SQL로 해석하여 MARK TABLE에서 WERKS 필드가 '1101' 인 값을 가져옴
- 3 | MARC TABLE에는 WERKS = '1101' 인 ROW는 2건으로 해당 DATA는 LOCAL BUFFER에 저장
- 4 | SQL의 SY-SUBRC EQ 0. 에서 DATABASE에서 값을 가져오는 구문이 성공하게 되면 EXIT 되어 프로그램이 DISPLAY 하는 값은 '1101'의 첫번째 값
- 5 | 다시 SQL을 실행하면, LOCAL BUFFER에 WERKS = '1101'인 데이터가 존재해 DATABASE에 접근하지 않고, LOCAL BUFFER의 DATA를 반환

## OPEN SQL OVERVIEW |

OPEN SQL은 DATABASE DATA를 조회하고 변경하는 등의 기능을 수행한다. 데이터 베이스 시스템 | ORACLE MSSQL MAX DB등 관계없이 SQL 결과와 ERROR MESSAGE 가 반환된다. ABAP DICTIONARY에서 생성된 테이블, 뷰에만 적용되며 CLUSTER, POOLED TABLE에서는 제한된 기능만을 수행할 수 있다. OPEN SQL은 기본적으로 CLIENT에 종속적이다.



그림과 같이 ABAP DICTIONARY는 독립적인 여러 개의 CLIENT로 구성할 수 있다. TABLE은 CLIENT INDEPENDENT OBJECT이며 ABAP TABLE에 MANDT 필드가 존재하지 않으면 TABLE DATA도 클라이언트 독립 속성을 가지게 된다. 즉 실제 물리적인 TABLE은 MANDT 필드에 CLIENT 정보를 포함하여 DATA가 저장된다. OPEN SQL의 수행 결과가 성공하면 시스템 변수 SY-SUBRC = 0을 반환하며 SY-DBCNT는 데이터 LINE 수를 반환한다.

## OPEN SQL 데이터 읽기 |

SELECT <RESULT>	조회하고자 하는 테이블 필드명을 나열하며, 한 건 또는 여러 라인을 조회 가능
INTO <TARGET>	SELECT에서 읽어온 DATA 변수 저장 변수를 ABAP 프로그램에서 사용
FROM <SOURCE>	SELECT 할 TABLE 지정 INTO 이전   이후 위치 모두 가능
WHERE <COND>	조회하고자 하는 DATA 조건 추가 가능
GROUP BY <FIELDS>	여러 라인의 결과를 그룹으로 지정해 하나의 결과 도출
HAVING <COND>	GROUP의 조건을 설정하는 WHERE 구문
ORDER BY <FIELDS>	조회된 DATA를 정렬   SORT

## OPEN SQL 데이터 읽기 | SELECT | SELECT <LINES> <COLUMNS> ...

SELECT 구문은 DATABASE TABLE에서 필요한 DATA를 읽어 온다. SELECT 구문은 두 가지 부분으로 나누어지는데 <LINES>는 하나 또는 여러 라인을 선택할 때 사용되며 하나의 라인일 때는 SINGLE 구문을 사용한다. <COLUMNS>는 테이블 칼럼을 기술한다.

### SINGLE LINE

## SELECT SINGLE <COLUMNS> ... WHERE | 모든 칼럼을 읽을 때는 \* 사용

DATABASE에서 하나의 라인 값을 읽어오기 할 경우 SINGLE을 사용한다. SINGLE 구문을 사용하면 DATA를 한 건만 가져오기 때문에 원하는 DATA의 조건을 정확하게 알고 있어야 한다. WHERE 조건에 유일한 KEY를 추가해야 하며 만약, WHERE 구문이 잘못되어 복수의 라인을 읽으면 임의의 라인을 반환한다.

### SEVERAL LINES

## SELECT [DISTINCT] <COLUMNS> ... WHERE

여러 라인 조회 시 SELECT 결과가 내부 TABLE에 저장된다. 이를 ABAP에서는 INTERNAL TABLE이라 부른다. INTERNAL TABLE은 ABAP 메모리에 생성되는 DATA를 저장할 수 있는 가상의 TABLE이다. DISTINCT를 사용하면 중복된 값을 제외한다.

## SELECT [DISTINCT] <COLUMNS> ... WHERE ... ENDSELECT.

INTO 구문의 결과가 저장되는 곳이 INTERNAL TABLE이 아닌 필드 또는 WORK AREA | STRUCTURE 일 경우 ENDSELECT를 사용한다. 이 구문은 하나의 값을 읽어 구조에 삽입하고, 조건에 해당하는 값을 모두 읽어 올 때까지 LOOP를 수행한다.

### AS | ALIAS

## SELECT <COLUMNS> [AS <ALIAS>]

AS 구문을 사용하여, 칼럼 명에 별명을 지정할 수 있다.

### 동적인 SELECT 구문

## SELECT <LINES> ( <ITAB> ) ...

SELECT 구문의 칼럼을 동적으로 선언할 수 있으며 동적 구문을 저장하는 STRUCTURE 은 최대 72 자 까지 가능하며 CHAR 타입으로 NULL인 경우 \*와 같은 구문이 된다.

## OPEN SQL 데이터 읽기 | INTO

SELECT 구문에서 조회한 결과 값을 변수 | TARGET AREA에 저장하는 기능을 수행한다.

### WORK AREA | 구조체

SELECT ... INTO [ CORRESPONDING FIELDS OF ] <WA>.

여러 칼럼의 한 라인만 조회하고자 할 경우 WORK AREA | 변수 구조체에 값을 할당한다. 애스터리스크 \* 기호를 사용하면 전체 칼럼의 값을 읽어오며, CORRESPONDING FIELDS OF 구문을 사용하면 한 번에 WORK AREA 의 동일 필드명에 값을 할당한다. \* 를 사용하는 경우, 개별 필드를 SELECT 하는 것보다 비효율적이며 테이블의 한 라인 전체 DATA를 DB에서 읽어오는 것이 아닌 일정 크기만큼 잘라 SELECT 결과를 반환하는 것으로 성능에 큰 영향을 미칠 수 있다. 따라서 SELECT \* 구문의 사용을 삼가는 것이 바람직하다.

### INTERNAL TABLE

SELECT ... INTO | APPENDING [ CORRESPONDING FIELDS OF ]  
TABLE <ITAB>  
[ PACKAGE SIZE <N> ] ...

여러 라인을 조회할 경우 INTERNAL TABLE을 사용한다. APPENDING은 INTERNAL TABLE에 추가로 INSERT 하며, INTO 는 INTERNAL TABLE의 DATA를 삭제한 다음 INSERT한다. PACKAGE SIZE는 INTERNAL TABLE에 몇 개의 라인을 추가할 것인가를 설정한다. PACKAGE SIZE 5의 경우 5개의 값을 여러 번 읽어와 테이블에 추가하며, 이때 ENDSELECT를 반드시 사용해야한다.

### SINGLE FIELD

SELECT ... INTO ( F1, F2, ... ) ...

SELECT CARR CONN INTO ( GV\_CARRID, GV\_CONNID ) FROM SFLIGHT.

TABLE의 개별 칼럼을 조회하거나 AGGREGATE 함수를 사용할 때는 위의 구문과 같이 사용한다. INTO 구문 다음에 두 개 이상의 TARGET이 필요한 경우에는 괄호와 변수명을 붙여 쓴다. 공백이 존재하면 SYNTAX ERROR가 발생한다. SELECT 구문에서 2개의 필드가 필요한 경우 2번째 구문과 같이 사용한다.

## OPEN SQL 데이터 읽기 | FROM | SELECT ... FROM TABLE OPTION ...

FROM 구문은 DATA를 SELECT 할 대상 테이블 또는 VIEW를 지정한다. FROM 구문 다음에는 하나의 TABLE을 지정하거나 여러 개의 TABLE을 JOIN 할 수 있다. ALIAS 를 사용하여 TABLE명에 별명을 붙일 수 있으며, 테이블을 동적으로 선언할 수 있다. FROM 구문은 TABLE을 정의하는 부분과 DB 접근을 컨트롤 하는 부분 OPTION으로 나뉘어진다.

CLIENT SPECIFIED	자동 CLIENT 설정을 해제
BYPASSING BUFFER	SAP LOCAL BUFFER에서 값을 읽지 않으며 BUFFERING 설정시에도 DATABASE TABLE에서 SELECT를 수행
UP TO N ROWS	SELECT의 ROW 개수를 제한하며 사용자 실수로 대량의 데이터 요청 시 DATABASE 성능 저하를 예방

## 정적인 TABLE 선택

**SELECT ... FROM <DBTAB> [AS <ALIAS>] <OPTIONS>**

하나의 테이블을 정적으로 선언할 때 사용한다. ALIAS를 사용할 수 있으며 이 경우 TABLE 명을 SELECT 구문에서 사용할 수 없다.

## 동적인 TABLE 선택

**SELECT ... FROM (DBTAB) .**

테이블 이름을 동적으로 선언하여 사용할 수 있다. 이때 TABLE 이름은 반드시 대문자로 지정해야 하며, ABAP DICTIONARY에 존재하는 이름이어야 한다.

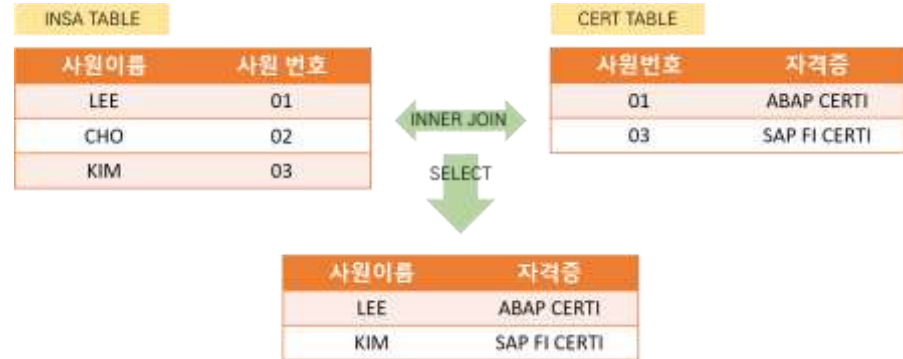
## JOIN

**SELECT ... FROM <TAB> [ INNER | OUTER (LEFT) ] JOIN <DBTAB> [AS <ALIAS>] ON <COND> <OPTIONS>...**

관계형 데이터베이스에서 여러 개의 테이블 값을 동시에 읽어 올 경우 JOIN을 사용하게 된다. 일반적으로 PRIMARY KEY 와 FOREIGN KEY를 사용하여 JOIN하는 경우가 대부분이나 때로는 논리적인 값들의 연관으로 JOIN 하는 경우도 있다. 두 테이블 간의 연결 조건은 ON 구문을 사용한다. JOIN에 사용되는 필드가 INDEX에 존재할 때 빠른 성능이 보장된다. ABAP 프로그램에서는 테이블 간의 JOIN을 하기 보다는 INTERNAL TABLE에 DATA를 저장하고 LOOP 구문을 이용하여, 추가 정보를 SELECT 하여 INTERNAL TABLE의 내용을 MODIFY 하는 경우가 많다. 그러나 이 경우 LOOP를 수행하며 DB에 반복 접근하며 프로그램을 느리게 만들기 때문에 JOIN이 가능한 경우 될 수 있다면 JOIN을 이용하는 것이 바람직하다. WHERE 조건과 ON 조건이 기술되어 있을 경우, WHERE 조건이 먼저 수행되고, 이 DATA를 기준으로 ON 조건에서 TABLE을 JOIN 한다.

## INNER JOIN 과 OUTER JOIN

아래 그림과 같이 인사 정보 TABLE 2개가 있다고 하자. INSA TABLE은 사원 기본 정보를 저장하며, CERT TABLE은 사원이 취득한 자격증 정보를 저장한다. 사원 LEE와 KIM 은 자격증을 가지나 사원 CHO는 자격증이 없다. 사원 번호 필드를 기준으로 INNER JOIN을 수행 시, CHO는 결과에서 제외된다.



기본적으로 모든 사원 정보는 조회하고, 자격증이 있는 경우에는 추가로 보여주고자 할 경우에는 OUTER JOIN을 사용한다. ABAP OPEN SQL 에는 LEFT OUTER JOIN만 사용할 수 있다. 이것은 JOIN 연결의 왼쪽에 있는 DATA를 기준으로 SELECT 결과가 도출된다.



## LINE 수 제한

SELECT ... FROM <TAB> UP TO <N> ROWS ...

다음 구문을 이용하여 TABLE에서 읽어오는 라인 수를 제한한다. 이는 리포트 프로그램의 조회 조건인 SELECTION SCREEN 에서 최대 적중 수를 제한하는 목적으로 많이 사용된다. 사용자가 조회 조건 값을 입력하지 않고 실행하는 경우 테이블의 모든 데이터를 읽기 때문에 서버에 부하를 주게 된다. 이럴 경우 최대 적중 수 조건을 기본으로 추가하여 LINE 수를 제한하게 된다.

OPEN SQL 읽기 | WHERE | SELECT ...WHERE <S> <OPERATOR> <F> ...

WHERE 조건은 SELECT 적중 수를 줄여 주고 사용자가 원하는 DATA를 정확하게 선택할 수 있도록 하는 조건이다. WHERE 조건에 사용되는 필드가 INDEX에서 사용될 때 빠른 성능을 보장한다. UPDATE DELETE 에서도 사용한다.

EQ   =	같다
NE   <>   ≠	같지 않다
LT   <	보다 작다
LE   <=	작거나 같다
GT   >	보다 크다
GE   >=	크거나 같다

## INTERVAL 조건

SELECT ... WHERE <S> [NOT] BETWEEN <F1> AND <F2>

INTERVAL 조건은 조건에 범위 값을 사용할 때 사용한다

## STRING 비교

SELECT ~ WHERE COL2 LIKE 'ABC%'

문자열을 비교할 때 LIKE 구문을 사용한다. 만약, ABC로 시작하는 ABCD, ABCE 와 같이 한자리만 비교할 경우 '-' 를 사용하여 WHERE COL2 LIKE 'ABC\_' 와 같이 표기한다.

## LIST VALUE

SELECT ... WHERE [NOT] IN ( <F1> ..., <FN> )

IN 구문을 사용하여, 여러 조건에 속한 경우의 값을 가져온다.

## SELECTION TABLE

SELECT ... WHERE <S> [NOT] IN <STAB>

IN 구문을 사용하여 SELECTION TABLE, RANGE 변수에 존재하는 값들을 조회할 수 있다. SELECTION TABLE, RANGE 변수는 INTERNAL TABLE과 유사하게 여러 ROW를 저장할 수 있는 변수 이다.

## DYNAMIC

SELECT ... WHERE ( <ITAB> ) ...

SELECT 구문의 조건을 설정하는 WHERE 구문을 동적으로 구성할 수 있다.

## FOR ALL ENTRIES

SELECT ... FOR ENTRIES IN <ITAB> WHERE <COND>

FOR ALL ENTRY 구문은 INTERNAL TABLE과 DB TABLE을 JOIN하는 개념과 유사하며, LOOP를 수행하면서 SQL을 수행한다. DB에 반복적으로 접근해 JOIN보다는 비효율적이나 ABAP에서 유용하게 활용된다. FOR ENTRIES를 사용할 경우 주의할 점은 ITAB의 칼럼과 비교 대상 TABLE의 칼럼 타입은 같아야 하며, LIKE, BETWEEN, IN 과 같은 비교 구문은 사용할 수 없다. ITAB의 중복된 값은 UNIQUE KEY값을 기준으로 하나만 남으며 ITAB이 NULL인 경우 모든 DATA를 읽는다. 만약 ITAB의 수가 3개이면 SELECT ~ END SELECT를 3번 수행하는 것과 동일 함으로 속도가 감소한다.

OPEN SQL 읽기 | GROUPING | \*동적 지정은 GROUP BY (ITAB) ...

SELECT <F1> <F2> <AGG> ... GROUP BY <F1> <F2> ...

AGGREGATE 함수를 사용하려면 SELECT 구문에 GROUP BY를 기술해야 한다. GROUP BY 구문은 TABLE의 특정 칼럼에 같은 값들이 존재할 때 이 값들의 정보를 요약하여 한 줄의 정보로 조회되게 한다. GROUP BY에 기술된 칼럼은 SELECT 구문에서 반드시 같은 칼럼으로 기술되어야 한다. SELECT 구문에서 칼럼과 AGGREGATE 함수를 사용할 수 있으며 종류는 다음과 같다.

AVG	평균
COUNT	개수
MAX	최댓값
MIN	최소값
STEDEV	표준편차
SUM	합계
VARIANCE	분산

## OPEN SQL 에서 SUM 을 사용하는 경우

SELECT CARRID SUM ( PRICE ) AS PRIE

FROM SFLIGHT

INTO CORRESPONDING FIELDS OF TABLE GT\_GLT

WHERE CARRID = 'AA'.

OPEN SQL 에서 SUM 을 이용하여 INTO CORRESPONDING FIELDS OF TABLE 구문을 사용하면 원하는 결과를 가져올 수 없다. 이때는 위의 예제와 같이 AS 구문을 이용해 별명 | ALIAS 을 지정해야 한다.

OPEN SQL 읽기 | HAVING |

SELECT <F1> <F2> <AGG> ... GROUP BY <F1> <F2> ... HAVING <COND>.

HAVING 구문은 GROUP BY로 조회한 SELECT 구문에 그룹의 조건을 추가한다. WHERE 조건 에서처럼 동적 DYNAMIC 선언이 가능하다.

## OPEN SQL 읽기 | SORT |

SELECT 결과로 조회된 DATA 가 ORDER BY에 기술된 칼럼 기준으로 정렬된다. ORDER BY 를 사용하지 않으면 임의로 정렬된 결과가 조회된다.

### ORDER BY PRIMARY KEY

SELECT <LINES> \* ... ORDER BY PRIMARY KEY.

ORDER BY PRIMARY KEY는 테이블 KEY에 의해 정렬되며, SELECT \* 구문인 경우에만 사용할 수 있다. JOIN 구문 및 VIEW에는 사용할 수 없다.

SELECT ... .... ORDER BY <F1> [ ASCENDING | DESCENDING ]

ORDER BY는 모든 칼럼을 사용할 수 있으며 ASCENDING, DESCENDING 구문으로 오름차순, 내림차순 정렬을 사용할 수 있다.

SELECT ... .... ORDER BY (ITAB)

ORDER BY 구문은 동적 선언이 가능하며 CHAR TYPE, 72자를 넘을 수 없다.

\*ORDER BY 구문에서는 AVG 와 같은 AGGREGATE 함수를 사용할 수 없으니 SELECT 구문에서 AS 명령어를 이용하여 사용한다.

## OPEN SQL 읽기 | SUBQUERY |

SUBQUERY는 SELECT 구문의 WHERE 조건에 또 다른 SELECT 구문을 추가하여 값을 제한하는 목적으로 사용된다.

### SCALAR SUBQUERY

SCALAR SUBQUERY는 SELECT 절 안에 기술된 SELECT 절로 하나의 행으로부터 하나의 칼럼 값 또는 AGGREGATE 함수만을 반환하는 SUBQUERY로서, JOIN 구문과 유사한 역할을 수행한다. 유의할 점은 다음과 같다

- 1 | SCALAR SUBQUERY는 반드시 한 칼럼만 반환한다
- 2 | SCALAR SUBQUERY는 NESTED LOOP 방식으로 처리된다
- 3 | SCALAR SUBQUERY가 실행되는 횟수는 ROW 이다
- 4 | 반복되는 코드나 마스터 유형의 테이블을 조회하는 경우 효율적이다

### NON-SCALAR SUBQUERY

SUBQUERY의 결과가 존재하면 TRUE를 반환하며, 존재하지 않으면 FALSE를 반환한다. EXISTS 구문을 이용하여 구현한다.



## OPEN SQL DATA 변경 | INSERT |

INSERT INTO <TARGET> <LINES>.  
<F2> [ ASCENDING | DESCENDING ] ...

TABLE에 하나 또는 여러 개의 DATA를 삽입한다. <TARGET> 은 TABLE 이름으로, 동적 선언할 수 있다. CLIENT를 지정할 수도 있으며, TABLE 이름을 정적 또는 동적으로 다음과 같이 선언할 수 있다.

INSERT INTO DBTAB [CLIENT SPECIFIED] <LINES>.  
INSERT INTO <NAME> [CLIENT SPECIFIED] <LINES>.

### SINGLE LINE

INSERT INTO <TARGET> VALUES <WA>.  
INSERT INTO <TARGET> FROM <WA>.

TABLE에 하나의 LINE을 삽입하기 위한 문장으로 <WA>는 TABLE과 같은 구조로 선언되어야 한다. 만약 INSERT <DBTAB> 구문을 쓰려면 TABLES: <DBTAB> 선언이 되어야 한다.

### SEVERAL LINES

INSERT <TARGET> FROM TABLE <ITAB>.  
[ ACCEPTING DUPLICATE KEYS ] .

INTERNAL TABLE의 모든 값을 한 번에 TABLE에 삽입한다. 같은 KEY 값을 INSERT 하게 되면, DUMP ERROR 가 발생하는데 이를 방지하기 위해, ACCEPTING DUPLICATE KEYS 구문을 사용한다. INSERT 구문이 실패하면

SY-SUBRC 값 4를 반환한다.

TABLE DATA 확인하기	
SE11	ABAP DICTIONARY 테이블과 같은 오브젝트를 조회 수정 생성하는 용도
SE16	DATA BROWSER 데이터를 조회하는 용도

## OPEN SQL DATA 변경 | UPDATE |

UPDATE INTO <TARGET> <LINES>.

TABLE의 하나 또는 여러 LINE을 변경한다. <TARGET> 은 TABLE 이름으로, 동적으로 선언할 수 있다. CLIENT를 지정할 수 있다.

UPDATE INTO <DBTAB> [CLIENT SPECIFIED] <LINES>.

### SINGLE LINE

UPDATE <TARGET> FROM <WA>.

WORK AREA <WA> 는 TABLE과 같은 구조로 선언되어야 하며, UPDATE <DBTAB> 구문을 쓰려면 TABLES: <DBTAB> 이 선언되어야 한다.

### SEVERAL LINES

INSERT <TARGET> FROM TABLE <ITAB>.

UPDATE <TARGET> SET <SET 1> <SET 2> WHERE <COND>.

INTERNAL TABLE의 여러 데이터를 한 번에 반영한다. SET 필드 1 필드 2 와 같이 필드를 여러 개 선언할 수 있다. SELECT 구문과 동일하게 WHERE 조건을 사용할 수 있다. WORK AREA 를 사용하는 것보다 UPDATE SET 구문을 이용해 개별 칼럼 값을 변경하는 것이 성능이 우수하다.

## OPEN SQL DATA 변경 | DELETE |

DELETE <TARGET> <LINES>.

TABLE의 하나 또는 여러 LINE을 삭제한다. <TARGET> 은 TABLE 이름으로, 동적으로 선언할 수 있다. CLIENT를 지정할 때는 동적 | 정적으로 선언할 수 있다.

DELETE [FROM] (<NAME>) [CLIENT SPECIFIED] <LINES>.

DELETE <DBTAB> [CLIENT SPECIFIED] <LINES>.

### SINGLE LINE

DELETE <TARGET> FROM <WA>.

WORK AREA <WA> 는 TABLE과 같은 구조로 선언되어야 하며, DELETE <DBTAB> 구문을 쓰려면 TBLES : <DBTAB> 이 선언되어야 한다.

### SEVERAL LINES

DELETE FROM <TARGET> WHERE <COND>.

INTERNAL TABLE의 WHERE 조건에 해당하는 모든 값을 한 번에 삭제한다

## OPEN SQL DATA 변경 | MODIFY |

MODIFY <TARGET> <LINES>.

MODIFY는 UPDATE 구문과 INSERT 구문을 합한 기능을 수행한다. KEY 값을 가지는 DATA 가 TABLE에 존재하면 UPDATE하고, 존재하지 않을 때에는 INSERT를 수행한다. TABLE에 하나 또는 여러 LINE을 UPDATE 또는 INSERT 할 수 있으며 <TARGET>는 TABLE 명으로 동적으로 선언할 수도 있다. CLIENT 지정시에는 TABLE 명을 정적 / 동적으로 선언할 수 있다.

MODIFY <DBTAB> [CLIENT SPECIFIED] <LINES>.

MODIFY <NAME> [CLIENT SPECIFIED] <LINES>.

### SINGLE LINE

MODIFY <TARGET> FROM <WA>.

WORK AREA <WA> 는 TABLE과 같은 구조로 선언되어야 하며, DELETE <DBTAB> 구문을 쓰려면 TBLES : <DBTAB> 이 선언되어야 한다.

### SEVERAL LINES

MODIFY <TARGET> FROM TABLE <ITAB>

INTERNAL TABLE의 모든 값을 한 번에 변경 또는 추가한다

SE80 | 0501 | 신호등 예제 - AA 17을 수행해보자

```
*SELECT F1 F2 F3  
*FROM DB TABLE  
*INTO TABLE INTERNAL TABLE  
*WHERE 조건  
*INTO CORRESPONDING FIELDS OF INTERNAL TABLE  
*LIKE와 TYPE의 차이는 ? DATA 로 선언된 OBJECT 이면 LIKE !  
  
*FUNCTION GROUP - FUNCTION MODULE  
*FUNCTION PROGRAM 이름은 앞에 SAPL 이 붙는다  
*SCREEN을 생성할 수 있다.  
*CLIENT SPECIFIED 는 클라이언트를 지정 사용할 때 사용한다.  
*신호등 예제 - AA 17
```

TYPE-POOLS:

ICON,  
COL.

CONSTANTS:

GC\_LIMIT\_RED TYPE ZS\_FLGHTOCC VALUE 98,  
GC\_LIMIT\_YELLOW TYPE ZS\_FLGHTOCC VALUE 75.

DATA:

GT\_FLIGHTS TYPE ZBC400\_T\_FLIGHTS,  
GS\_FLIGHT TYPE ZBC400\_S\_FLIGHT.

PARAMETERS:

PA\_CAR TYPE ZBC400\_S\_FLIGHT-CARRID.

SELECT-OPTIONS:

SO\_CON FOR GS\_FLIGHT-CONNID.

TRY.

CALL METHOD ZCL\_BC400\_FLIGHTMODEL=>GET\_FLIGHTS\_RANGE

EXPORTING

IV\_CARRID = PA\_CAR

IT\_CONNID = SO\_CON[]

IMPORTING

ET\_FLIGHTS = GT\_FLIGHTS.

CATCH ZCX\_BC400\_NO\_DATA .

WRITE: / 'NO FLIGHTS FOR SELECTED FLIGHT CONNECTION'

.

WRITE:/ TEXT-

NON. "ALTERNATIVE USAGE OF TEXT SYMBOL "

ENDTRY.

LOOP AT GT\_FLIGHTS INTO GS\_FLIGHT.

NEW-LINE.

IF GS\_FLIGHT-PERCENTAGE >= GC\_LIMIT\_RED.

```

WRITE ICON_RED_LIGHT AS ICON.
ELSEIF GS_FLIGHT-PERCENTAGE >= GC_LIMIT_YELLOW.
WRITE ICON_YELLOW_LIGHT AS ICON.
ELSE.
WRITE ICON_GREEN_LIGHT AS ICON.
ENDIF.

```

```

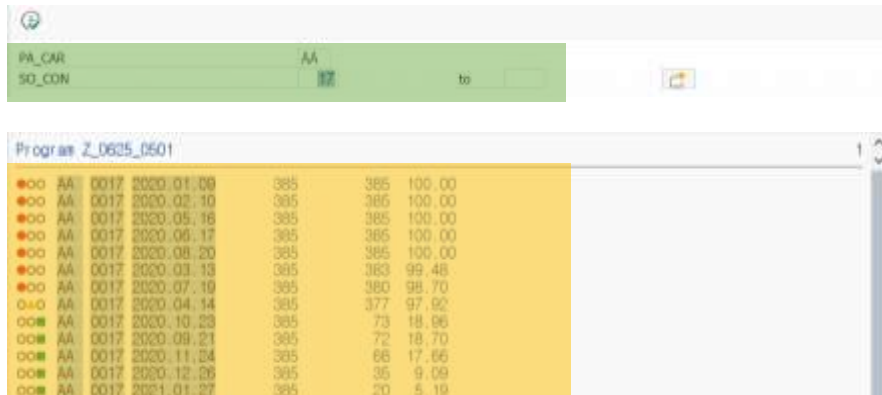
WRITE: GS_FLIGHT-CARRID COLOR COL_KEY,
       GS_FLIGHT-CONNID COLOR COL_KEY,
       GS_FLIGHT-FLDATE COLOR COL_KEY,
       GS_FLIGHT-SEATSMAX,
       GS_FLIGHT-SEATSOCC,
       GS_FLIGHT-PERCENTAGE.

```

```

ENDLOOP.

```



The screenshot shows the SAP program Z\_0625\_0501. At the top, there is a selection screen with fields for PA\_CAR (containing 'AA') and SO\_CON (containing '123'). Below this, a data table is displayed with the following columns: CARRID, CONNID, FLDATE, SEATSMAX, SEATSOCC, and PERCENTAGE. The table contains 18 rows of data.

CARRID	CONNID	FLDATE	SEATSMAX	SEATSOCC	PERCENTAGE
AA	0017	2020.01.09	385	385	100.00
AA	0017	2020.02.10	385	385	100.00
AA	0017	2020.05.16	385	385	100.00
AA	0017	2020.06.17	385	385	100.00
AA	0017	2020.08.20	385	385	100.00
AA	0017	2020.03.13	385	383	99.48
AA	0017	2020.07.19	385	380	98.70
AA	0017	2020.04.14	385	377	97.92
AA	0017	2020.10.23	385	73	18.96
AA	0017	2020.09.21	385	72	18.70
AA	0017	2020.11.24	385	66	17.66
AA	0017	2020.12.26	385	35	9.09
AA	0017	2021.01.27	385	20	5.19

## SE80 | 0502 | 이벤트 프로그램 구조에 대해 알아보기

*\*EVENT로 구성되는 프로그램 / 프로그램 생성시 EXECUTABLE PROGRAM*

*\*순서대로 진행되기 때문에 절차적 언어라고 한다.*

*\*PROGRAM START = REPORT...*

*\*INITIALIZATION*

*\*AT SELECTION-SCREEN*

*\*START OF SELECTION*

*\*DISPLAY LISR BUFFER AS LIST*

*\*실제 순서가 지정되어 있어 프로그램상의 나열은 중요하지 않으나 가독성을 위해 정확한 나열을 필요로 한다.*

**PARAMETERS:** PA\_CAR TYPE S\_CARR\_ID.

**INITIALIZATION.**

*\*로직 수행시 사용되는 초기값을 지정한다. / 화면 송출시 표시되는 값*

*\*SELECTION SCREEN 화면이 열리기 전에 화면 필드 값을 초기화 하는데 주로 사용된다.*

*\*필드를 초기화 하며 DEFAULT 값을 세팅한다.*

*\*즉 사용자가 자주 사용하는 값을 자동으로 입력되게 한다.*

PA\_CAR = 'LH'.

**AT SELECTION-SCREEN.**

*\*INPUT CHECK / 입력화면 제어*

\*만약 AT SELECTION-SCREEN 에서 ERROR MESSAGE 송출 시 이전 화면으로 돌아 감

\*사용자가 SELECTION SCREEN에 값을 입력하기 전/후에 작동한다.

\*SELECTION-SCREEN 에서 INPUT FIELD 값이 변동 되었을 때 실행되는 이벤트

\*사용자 액션에 대해 반응하고 화면 필드를 조절하다.

## START-OF-SELECTION.

\*START OF SELECTION 에서는 SQL 및 가공

\*START OF SELECTION 에서 ERROR MESSAGE 송출 시 PROGRAM 종료

\*사용자가 F8 실행 버튼을 클릭하면 LDB 데이터베이스에서 값을 읽어온다.

\*일반적으로 SELECT 수문이 사용되는 블록으로 LDB를 사용한 REPORT 에서 는 GET TABLE이 사용된다.

\*조회 화면의 필드에 대한 초기 값 세팅 및 데이터 검증 VALIDATION이 완료되면 DATA를 가져오는 작업 수행

\*LDB를 이용한 프로그램은 GET 구문을 , 일반 프로그램이라면 SQL 구문을 수행한다.

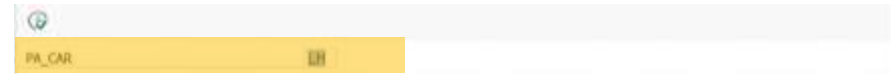
\*시간이 지연될 경우 SAPGUI\_PROGRESS\_INDICATOR 함수를 이용해 모래시계를 보여줄 수 있다.

## END-OF-SELECTION.

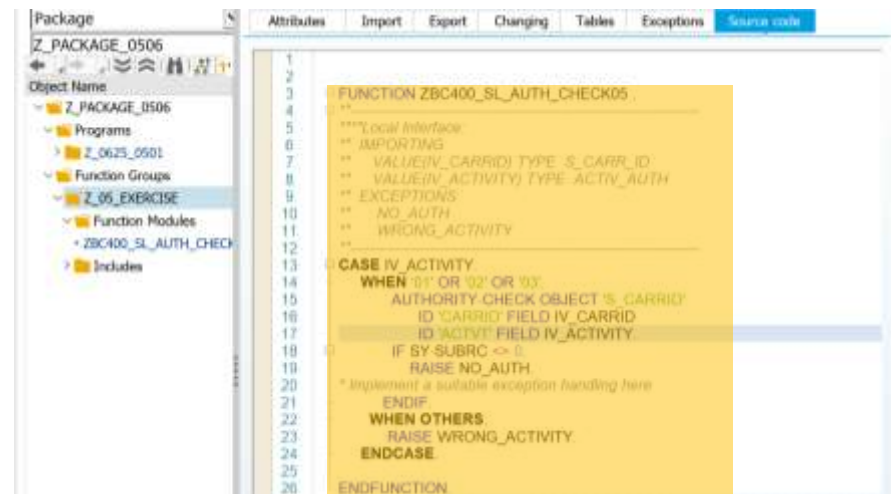
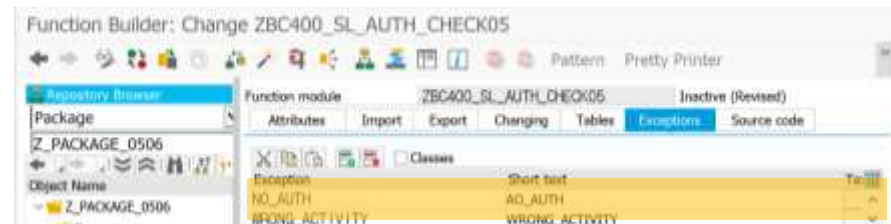
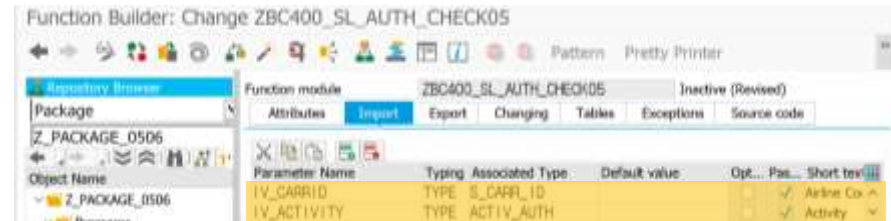
\*결과 화면

\*실행 환경에서 호출되는 마지막 이벤트로서 SELECT 구문에서 모든 DATA를 읽은 후 화면에 WRITE 하기 전 수행된다.

\*INTERNAL TABLE에 저장된 DATA들을 변형하는 작업을 할 수 있다.



SE80 | 0503 | FUNCTION | ZBC400\_SL\_AUTH\_CHECK05를 생성하자!



SU21 | 0503 | FUNCTION | 권한 설정을 해보자!

Maintain Authorization Objects

Regenerate SAP\_ALL

Class/Object	Text	Type
• ZCD6	Object Class for DEMO CDS VIEW	Object ...
• ZD08	Demo Dash Board Objects	Object ...
• ZT00	zauth test	Object ...
• Z_AUTH_00	Z_auth_00	Author...

SU21 | 0503 | CREATE AUTHORIZATION

Create Authorization Object

Object	Z_LH_05
Text	LH ONLY
Class	ZT00 zauth test
Author	

Authorization fields

Authorization Field	Short Description...
CARRID	Airline Code
ACTVT	Activity

SU21 | 0503 | MAINTAIN AUTHORIZATION

Maintain Authorization Object

Object	Z_LH_05
Text	LH ONLY
Class	ZT00 zauth test
Author	CLSAP05

Authorization fields

Authorization Field	Short Description...
CARRID	Airline Code
ACTVT	Activity

Define Values

Object	Z_LH_05	LH ONLY
Field name	ACTVT	Activity

Activities

S...	A...	Text
✓ 01	Add or Create	
✓ 02	Change	
✓ 03	Display	

PFCG | 0503 | SINGLE ROLE

Create Roles

Other role Inheritance

Role Z\_LH\_05

Description LH ONLY

Target System

Administration Information

User Created

Date

Time 00:00:00

Transaction Inheritance

Derive from Role

Change Roles

Other role Inheritance

Role Z\_LH\_05

Description LH ONLY

Target System No destination

Description Menu Workflow Authorizations User MiniApps Personalization

Created

User

Date

Time 00:00:00

Last Changed

User

Date

Time 00:00:00

Last Profile Generation

User

Date

Time 00:00:00

Information About Authorization Profile

Profile Name

Profile Text

Status No authorization data exists



Choose Template

Template	Text for Template
/IWBEP/GW_SUPPORT_RO	Read-only role for SAP Gateway supportability
/IWBEP/RT_BEP_ADM	Role Template for Backend Event Publisher Administrator
/IWBEP/RT_BEP_USR	Role template for backend event publisher users
/IWBEP/RT_MGW_ADM	Role Template for OData Channel Administrator
/IWBEP/RT_MGW_DEV	Role Template for OData Channel Developer
/IWBEP/RT_MGW_USR	Role Template for OData Channel User
/IWBEP/RT_SUB_USR	Role Template for On-behalf Subscription User
/IWBEP/RT_USS_ADMUSR	Admin User - SAP Gateway User Self Service
/IWBEP/RT_USS_INTUSR	Internet User - SAP Gateway User Self Service
/IWBEP/RT_USS_SRVUSR	Service User - SAP Gateway User Self Service
/IWFND/GW_SUPPORT_RO	Read-only role for SAP Gateway supportability

Change Role: Authorizations

Selection criteria Manually Organizational levels... Trace Information

Role Z\_LH\_05

Status Edit Search Values

Group/Object/Authorization/Field	Maintenance ...	A...	Value	Text
Object class ZT00	Manually			zauth test
Authorization Object Z_LH_05	Manually			LH ONLY
Authorization T-0500062600	Manually			LH ONLY
CARRID	Manually			Airline Code
ACTVT	Manually			Activity

Change Role: Insert Authorizations

Insert chosen

Choose the authorizations you want to insert

Integration Orchestrator	Y10
zauth test1	ZAUI
Object Class for DEMO CDS VIEW	ZCOS
Demo Dash Board Objects	ZDDB
zauth test	ZT00
z_auth_00	Z_AUTH_00
lh only	Z_LH_00
LH ONLY	Z_LH_001
lh only	Z_LH_01
lh only	Z_LH_02
lh only	Z_LH_03
lh only	Z_LH_04
LH ONLY	Z_LH_05
Airline Code	CARRID
Activity	ACTVT
+ LH ONLY	Z_LH_05
Airline Code	CARRID
Activity	ACTVT

Change Role: Authorizations

Selection criteria Manually Organizational levels... Trace Information

Role Z\_LH\_05

Status Edit Search Values

Group/Object/Authorization/Field	Maintenance ...	A...	Value	Text
Object class ZT00	Manually			zauth test
Authorization Object Z_LH_05	Manually			LH ONLY
Authorization T-0500062600	Manually			LH ONLY
CARRID	Manually			Airline Code
ACTVT	Manually			Activity

Field values

Object Z\_LH\_05 LH ONLY

Field Name CARRID Airline Code

Full authorization

Value Intrvl

'From' 'To'

LH

Change Role: Authorizations

Selection criteria Manually Organizational levels... Trace Information

Role Z\_LH\_05

Status Edit Search Values

Group/Object/Authorization/Field	Maintenance ...	A...	Value	Text
Object class ZT00	Manually			zauth test
Authorization Object Z_LH_05	Manually			LH ONLY
Authorization T-0500062600	Manually			LH ONLY
CARRID	Manually			Airline Code
ACTVT	Manually			Activity

Field name ACTVT Activity

Full authorization

Activities

S...	A...	Text
<input type="checkbox"/>	01	Add or Create
<input type="checkbox"/>	02	Change
<input checked="" type="checkbox"/>	03	Display

Change Role: Authorizations

Selection criteria Manually Organizational levels... Trace Information

Role Z\_LH\_05

Status Edit

Group/Object/Authorization/Field

Group/Object/Authorization/Field	Maintenance ... A...	Value	Text
Object class ZT00	Manually		auth test
Authorization Object Z_LH_05	Manually		LH ONLY
Authorization: T-DS08062600	Manually		LH ONLY
CARRJD	Manually	LH	Airline Code
ACTVT	Manually	Display	Activity

Assign Profile Name for Generated Authorization Profile

You can change the default profile name here

Profile name T-DS080626

Text Profile for role Z\_LH\_05

Save Without Profile Name

PFCG | 0503 | 뒤로 가기

Other role Inheritance Information

Role Z\_LH\_05

Description LF ONLY

Target System No destination

Description Menu Workflow Authorizations Use MiniApps Personalization

User Assignments

User ID	User Name	From	to
CLSAP05	CLSAP05	2020.06.25	9999.12.31

Attempted Profile Comparison

User	Date	Time
		00:00:00

Profile comparison successful

User	Date	Time
		00:00:00

Information for user master comparison

Status User assignments for generated profiles are not current

Complete comparison Information

Compare Role User Master Record

Attempted Profile Comparison

User	Date	Time
CLSAP05	2020.06.25	11:03:09

Profile comparison successful

User	Date	Time
CLSAP05	2020.06.25	11:03:09

Information for user master comparison

Status Comparison of user master record completed

Change Roles

Other role Inheritance Information

Role Z\_LH\_05

Description LF ONLY

Target System No destination

Description Menu Workflow Authorizations Use MiniApps Personalization

User Assignments

User ID	User Name	From	to
CLSAP05	CLSAP05	2020.06.25	9999.12.31

SE80 | 0503 | FUNCTION | 권한 설정 확인해보기

Exception WRONG\_ACTIVITY

Import parameters	Value
IV_CARRID	AAA
IV_ACTIVITY	00

Runtime: 1.985 Microseconds

Import parameters	Value
IV_CARRID	AA
IV_ACTIVITY	03



SE80 | 0503 | 0501 신호등 예제 - AA 17를 변형하여 수행해보자

TYPE-POOLS:

ICON,  
COL.

CONSTANTS:

GC\_LIMIT\_RED TYPE ZS\_FLGHTOCC VALUE 98,  
GC\_LIMIT\_YELLOW TYPE ZS\_FLGHTOCC VALUE 75,  
GC\_ACTVT\_DISPLAY TYPE ACTIV\_AUTH VALUE '03'.

DATA:

GT\_FLIGHTS TYPE ZBC400\_T\_FLIGHTS,  
GS\_FLIGHT TYPE ZBC400\_S\_FLIGHT.

PARAMETERS:

PA\_CAR TYPE ZBC400\_S\_FLIGHT-CARRID.

SELECT-OPTIONS:

SO\_CON FOR GS\_FLIGHT-CONNID.

INITIALIZATION.

PA\_CAR = 'LH'.  
SO\_CON-SIGN = 'I'.  
SO\_CON-OPTION = 'EQ'.  
SO\_CON-LOW = '0017'.

APPEND SO\_CON.

CLEAR SO\_CON.

SO\_CON-SIGN = 'I'.  
SO\_CON-OPTION = 'EQ'.  
SO\_CON-LOW = '0020'.

APPEND SO\_CON.

AT SELECTION-SCREEN.

TRY.

CALL METHOD ZCL\_BC400\_FLIGHTMODEL=>CHECK\_AUTHORITY

EXPORTING

IV\_CARRID = PA\_CAR  
IV\_ACTIVITY = GC\_ACTVT\_DISPLAY.

CATCH ZCX\_BC400\_NO\_AUTH .

MESSAGE E046(BC400) WITH PA\_CAR.

ENDTRY.

START-OF-SELECTION.

TRY.

CALL METHOD ZCL\_BC400\_FLIGHTMODEL=>GET\_FLIGHTS\_RANGE

EXPORTING

IV\_CARRID = PA\_CAR  
IT\_CONNID = SO\_CON[] "BRACKETS NEEDED FOR SO\_CON"

IMPORTING

ET\_FLIGHTS = GT\_FLIGHTS.

CATCH ZCX\_BC400\_NO\_DATA .

WRITE / 'NO FLIGHTS FOR THE SELECTED FLIGHT CONNECTION'.

ENDTRY.

LOOP AT GT\_FLIGHTS INTO GS\_FLIGHT.

NEW-LINE.

IF GS\_FLIGHT-PERCENTAGE >= GC\_LIMIT\_RED.

WRITE ICON\_RED\_LIGHT AS ICON.

ELSEIF GS\_FLIGHT-PERCENTAGE >= GC\_LIMIT\_YELLOW.

WRITE ICON\_YELLOW\_LIGHT AS ICON.

ELSE.

WRITE ICON\_GREEN\_LIGHT AS ICON.

ENDIF.

WRITE: /

GS\_FLIGHT-CARRID COLOR COL\_KEY,

GS\_FLIGHT-CONNID COLOR COL\_KEY,

GS\_FLIGHT-FLDATE COLOR COL\_KEY,

GS\_FLIGHT-SEATSMAX,

GS\_FLIGHT-SEATSOCC,

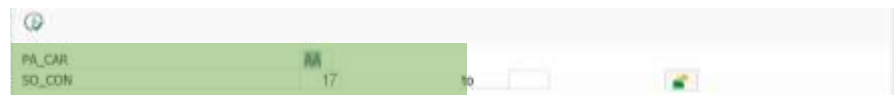
GS\_FLIGHT-PERCENTAGE.

ENDLOOP.



Report: Z\_D025\_0503

NO FLIGHTS FOR THE SELECTED FLIGHT CONNECTION



Report: Z\_D025\_0503

•••	AA:	0017	2020:01:09	385	385	100.00
•••	AA:	0017	2020:02:10	385	385	100.00
•••	AA:	0017	2020:05:16	385	385	100.00
•••	AA:	0017	2020:06:17	385	385	100.00
•••	AA:	0017	2020:08:20	385	385	100.00
•••	AA:	0017	2020:08:18	385	383	99.48
•••	AA:	0017	2020:07:19	385	380	98.70
•••	AA:	0017	2020:04:14	385	377	97.92
•••	AA:	0017	2020:10:23	385	73	18.96
•••	AA:	0017	2020:09:21	385	72	18.70
•••	AA:	0017	2020:11:24	385	68	17.66
•••	AA:	0017	2020:12:26	385	35	9.09
•••	AA:	0017	2021:01:27	385	20	5.19

SE80 | 0504 | SQL 구문을 사용해보자

DATA GT\_SF1 TYPE TABLE OF SFLIGHT.

SELECT \*

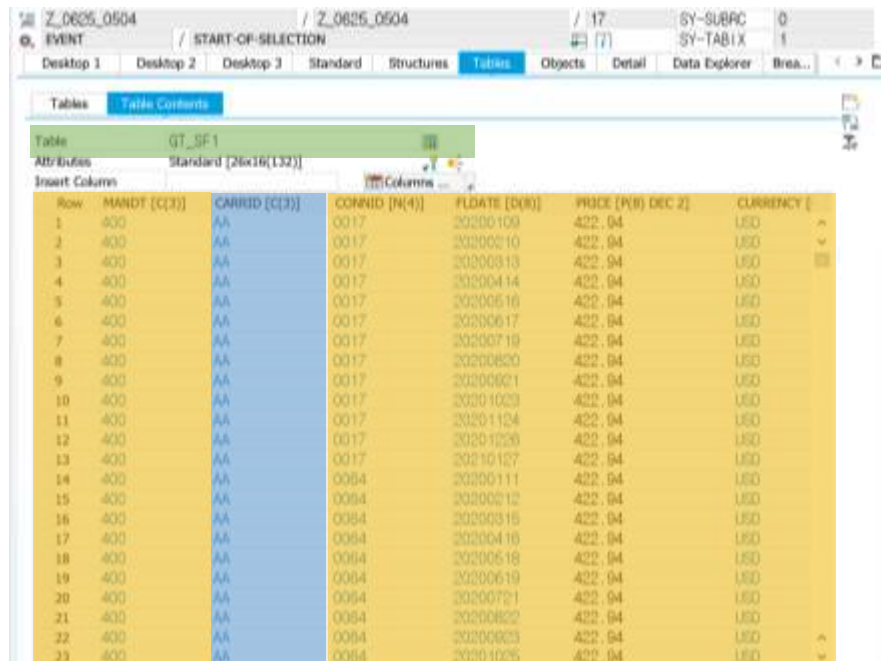
FROM SFLIGHT

INTO TABLE GT\_SF1

WHERE CARRID = 'AA'

ORDER BY PRIMARY KEY.

BREAK-POINT.



Row	MANDT	CARRID	CONNID	FLDATE	PRICE	CURRENCY
1	400	AA	0017	20200109	422.94	USD
2	400	AA	0017	20200210	422.94	USD
3	400	AA	0017	20200313	422.94	USD
4	400	AA	0017	20200414	422.94	USD
5	400	AA	0017	20200516	422.94	USD
6	400	AA	0017	20200617	422.94	USD
7	400	AA	0017	20200719	422.94	USD
8	400	AA	0017	20200820	422.94	USD
9	400	AA	0017	20200921	422.94	USD
10	400	AA	0017	20201023	422.94	USD
11	400	AA	0017	20201124	422.94	USD
12	400	AA	0017	20201226	422.94	USD
13	400	AA	0017	20210127	422.94	USD
14	400	AA	0064	20200111	422.94	USD
15	400	AA	0064	20200212	422.94	USD
16	400	AA	0064	20200315	422.94	USD
17	400	AA	0064	20200416	422.94	USD
18	400	AA	0064	20200518	422.94	USD
19	400	AA	0064	20200619	422.94	USD
20	400	AA	0064	20200721	422.94	USD
21	400	AA	0064	20200822	422.94	USD
22	400	AA	0064	20200923	422.94	USD
23	400	AA	0064	20201025	422.94	USD

SE80 | 0505 | SQL 구문을 사용해보자

DATA GT\_SF1 TYPE TABLE OF SFLIGHT.

SELECT \*

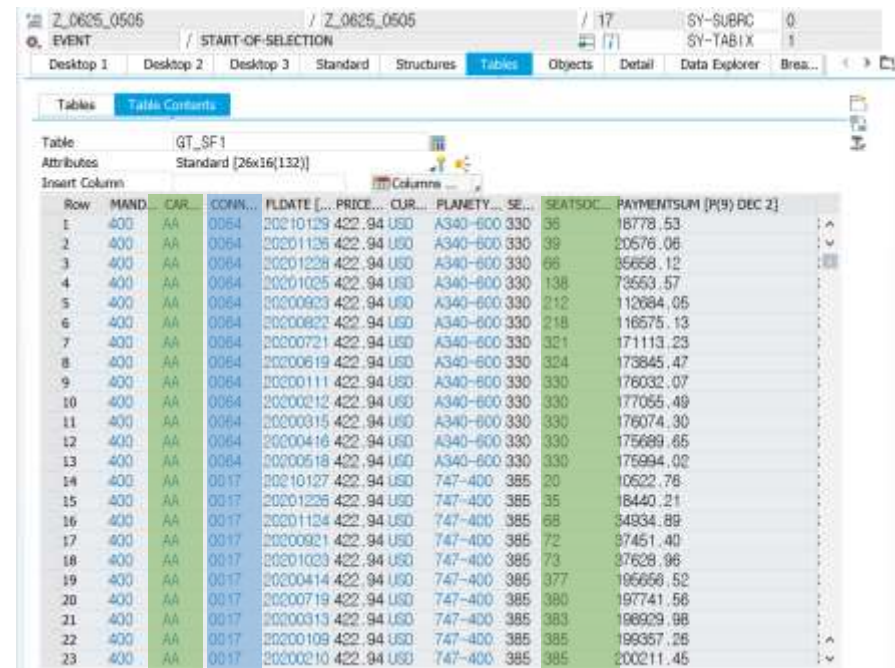
FROM SFLIGHT

INTO TABLE GT\_SF1

WHERE CARRID = 'AA'

ORDER BY CONNID DESCENDING SEATSOCC ASCENDING.

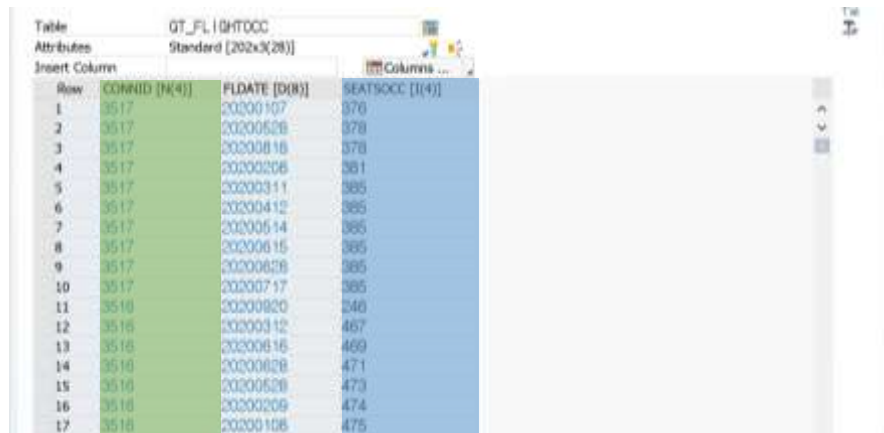
BREAK-POINT.



Row	MAND	CAR	CONN	FLDATE	PRICE	CUR	PLANET	SE	SEATSOCC	PAYMENTSUM
1	400	AA	0064	20210129	422.94	USD	A340-600	330	35	18778.53
2	400	AA	0064	20201126	422.94	USD	A340-600	330	39	20576.06
3	400	AA	0064	20201228	422.94	USD	A340-600	330	66	35658.12
4	400	AA	0064	20201025	422.94	USD	A340-600	330	138	73553.57
5	400	AA	0064	20200923	422.94	USD	A340-600	330	212	112684.05
6	400	AA	0064	20200822	422.94	USD	A340-600	330	218	116575.13
7	400	AA	0064	20200721	422.94	USD	A340-600	330	321	171113.23
8	400	AA	0064	20200619	422.94	USD	A340-600	330	324	173845.47
9	400	AA	0064	20200111	422.94	USD	A340-600	330	330	176032.07
10	400	AA	0064	20200212	422.94	USD	A340-600	330	330	177055.49
11	400	AA	0064	20200315	422.94	USD	A340-600	330	330	176074.30
12	400	AA	0064	20200416	422.94	USD	A340-600	330	330	175689.65
13	400	AA	0064	20200518	422.94	USD	A340-600	330	330	175994.02
14	400	AA	0017	20210127	422.94	USD	747-400	385	20	10522.76
15	400	AA	0017	20201226	422.94	USD	747-400	385	35	18440.21
16	400	AA	0017	20201124	422.94	USD	747-400	385	68	34934.89
17	400	AA	0017	20200921	422.94	USD	747-400	385	72	37451.40
18	400	AA	0017	20201023	422.94	USD	747-400	385	73	37628.96
19	400	AA	0017	20200414	422.94	USD	747-400	385	377	196656.52
20	400	AA	0017	20200719	422.94	USD	747-400	385	380	197741.56
21	400	AA	0017	20200313	422.94	USD	747-400	385	383	198929.98
22	400	AA	0017	20200109	422.94	USD	747-400	385	385	199357.25
23	400	AA	0017	20200210	422.94	USD	747-400	385	385	200211.45

SE80 | 0506 | SQL 구문을 사용해보자

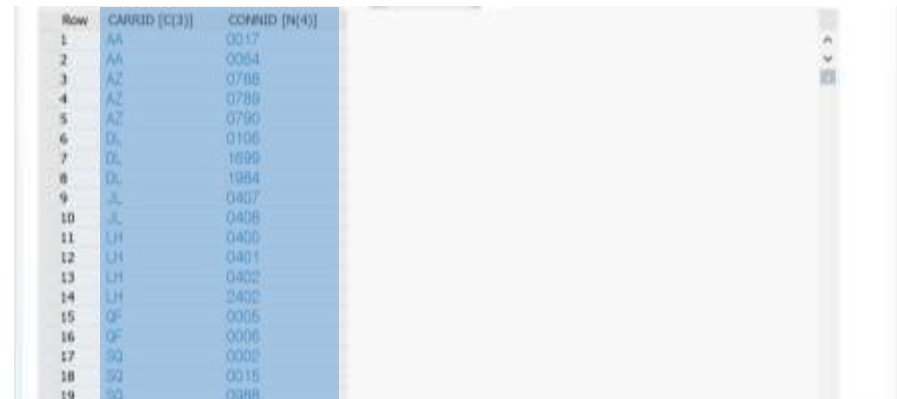
```
TYPES: BEGIN OF GTY_S_FLIGHTOCC,  
        CONNID TYPE SFLIGHT-CONNID,  
        FLDATE TYPE SFLIGHT-FLDATE,  
        SEATSOCC TYPE SFLIGHT-SEATSOCC,  
        END OF GTY_S_FLIGHTOCC.  
  
DATA GT_FLIGHTOCC TYPE TABLE OF GTY_S_FLIGHTOCC.  
  
SELECT CONNID FLDATE SEATSOCC  
        FROM SFLIGHT  
        INTO TABLE GT_FLIGHTOCC  
        WHERE SEATSOCC > 200  
        ORDER BY CONNID DESCENDING SEATSOCC ASCENDING.  
BREAK-POINT.
```



Row	CONNID [N(4)]	FLDATE [D(8)]	SEATSOCC [I(4)]
1	3517	20200107	376
2	3517	20200528	378
3	3517	20200818	378
4	3517	20200206	381
5	3517	20200311	385
6	3517	20200412	385
7	3517	20200514	385
8	3517	20200615	385
9	3517	20200628	385
10	3517	20200717	385
11	3518	20200820	240
12	3518	20200312	467
13	3518	20200616	469
14	3518	20200628	471
15	3518	20200528	473
16	3518	20200209	474
17	3518	20200106	475

SE80 | 0507 | SQL 구문을 사용해보자

```
TYPES: BEGIN OF GTY_S_FLIGHT,  
        CARRID TYPE SFLIGHT-CARRID,  
        CONNID TYPE SFLIGHT-CONNID,  
        END OF GTY_S_FLIGHT.  
  
DATA GT_FLIGHT TYPE TABLE OF GTY_S_FLIGHT.  
DATA GS_FLIGHT TYPE GTY_S_FLIGHT.  
  
SELECT DISTINCT CARRID CONNID  
        FROM SFLIGHT  
        INTO TABLE GT_FLIGHT  
        WHERE SEATSOCC > 200.  
BREAK-POINT.
```



Row	CARRID [C(3)]	CONNID [N(4)]
1	AA	0017
2	AA	0064
3	AZ	0789
4	AZ	0789
5	AZ	0790
6	DL	0106
7	DL	1699
8	DL	1984
9	JL	0407
10	JL	0408
11	LH	0400
12	LH	0401
13	LH	0402
14	LH	2402
15	OF	0005
16	OF	0006
17	SQ	0002
18	SQ	0015
19	SQ	0988

SE80 | 0508 | SQL 구문을 사용해보자

```
TYPES: BEGIN OF GTY_S_FLIGHT,  
        CONNID TYPE SFLIGHT-CONNID,  
        FLDATE TYPE SFLIGHT-FLDATE,  
        SEATSOCC TYPE SFLIGHT-SEATSOCC,  
        END OF GTY_S_FLIGHT.  
  
DATA GT_FLIGHT TYPE TABLE OF GTY_S_FLIGHT.  
DATA GS_FLIGHT TYPE GTY_S_FLIGHT.  
  
SELECT CONNID FLDATE SEATSOCC  
        FROM SFLIGHT  
        INTO TABLE GT_FLIGHT  
        WHERE SEATSOCC > 200  
        ORDER BY CONNID DESCENDING FLDATE.
```

```
LOOP AT GT_FLIGHT INTO GS_FLIGHT.  
        WRITE: / GS_FLIGHT-CONNID,  
                GS_FLIGHT-FLDATE,  
                GS_FLIGHT-SEATSOCC.  
ENDLOOP.
```

CONNID	FLDATE	SEATSOCC
3517	2020.01.07	376
3517	2020.02.09	391
3517	2020.03.11	385
3517	2020.04.12	385
3517	2020.05.14	385
3517	2020.05.29	378
3517	2020.06.15	385
3517	2020.06.29	385
3517	2020.07.17	385
3517	2020.08.18	378
3516	2020.01.08	475
3516	2020.02.09	474
3516	2020.03.12	467
3516	2020.04.13	475
3516	2020.05.15	475
3516	2020.05.29	473
3516	2020.06.16	469
3516	2020.06.29	471
3516	2020.07.18	475
3516	2020.08.19	475
3516	2020.09.20	240
3504	2020.01.11	475
3504	2020.02.12	474
3504	2020.03.15	475
3504	2020.04.18	475
3504	2020.05.18	471
3504	2020.05.29	470
3504	2020.06.19	475
3504	2020.06.29	470
3504	2020.07.21	475
3504	2020.08.22	270
2402	2020.01.12	472
2402	2020.02.13	470
2402	2020.03.16	475
2402	2020.04.17	475
2402	2020.05.19	475
2402	2020.05.29	471
2402	2020.06.20	469
2402	2020.06.29	467
2402	2020.07.22	475
2402	2020.09.24	271

SE80 | 0509 | SQL 구문을 사용해보자

```
TYPES: BEGIN OF GTY_S_FLIGHTOCC,  
        CNTALL TYPE I,  
        MINOCC TYPE SFLIGHT-SEATSOCC,  
        MAXOCC TYPE SFLIGHT-SEATSOCC,  
        SUMOCC TYPE SFLIGHT-SEATSOCC,  
        END OF GTY_S_FLIGHTOCC.
```

```
DATA GS_FLIGHTOCC TYPE GTY_S_FLIGHTOCC.
```

```
SELECT COUNT(*) MIN( SEATSOCC ) MAX( SEATSOCC ) SUM( SEATSOCC )  
        FROM SFLIGHT  
        INTO GS_FLIGHTOCC.
```

```
WRITE: / GS_FLIGHTOCC-CNTALL,  
        GS_FLIGHTOCC-MINOCC,  
        GS_FLIGHTOCC-MAXOCC,  
        GS_FLIGHTOCC-SUMOCC.
```

Report Z_0825_0509			
358	0	475	87,582

SE80 | 0510 | SQL 구문을 사용해보자

```
TYPES: BEGIN OF GTY_S_FLIGHTOCC,  
        CNTALL TYPE I,  
        CNTCON TYPE I,  
        SUMOCC TYPE SFLIGHT-SEATSOCC,  
        END OF GTY_S_FLIGHTOCC.
```

```
DATA GS_FLIGHTOCC TYPE GTY_S_FLIGHTOCC.
```

```
SELECT COUNT(*) COUNT( DISTINCT CONNID ) SUM( DISTINCT SEATSOCC )  
        FROM SFLIGHT  
        INTO GS_FLIGHTOCC.
```

```
WRITE: / GS_FLIGHTOCC-CNTALL,  
        GS_FLIGHTOCC-CNTCON,  
        GS_FLIGHTOCC-SUMOCC.
```

Report Z_0825_0510			1
358	26	23,412	

SE80 | 0511 | SQL 구문을 사용해보자

DATA COUNT TYPE I.

```
SELECT COUNT(*)  
    FROM SFLIGHT  
    WHERE CARRID = 'AA'  
    AND CONNID = '0017'  
    AND FLDATE = '20090101'.
```

BREAK-POINT.

```
IF SY-SUBRC = 0.  
ENDIF.
```

SE80 | 0512 | SQL 구문을 사용해보자

\* \_ PLACE POSITION  
\*% ANY POSITION  
\*~ 필드내에서 비교할 시 참조 사용

```
DATA: BEGIN OF GS_STR,  
    SUM TYPE SBOOK-FORCURAM,  
    FORCURKEY TYPE SBOOK-FORCURKEY,  
    END OF GS_STR.
```

DATA GT\_ITAB LIKE TABLE OF GS\_STR.

*\*SELECT 문에 , 면 @ , 없으면 안써도 됩니다.*

```
SELECT SUM( FORCURAM ) AS SUM, FORCURKEY  
    FROM SBOOK  
    INTO TABLE @GT_ITAB  
    WHERE CUSTOMID = '00000144'  
    GROUP BY FORCURKEY.
```

LOOP AT GT\_ITAB INTO GS\_STR.

```
    WRITE:/ GS_STR-SUM, GS_STR-FORCURKEY.  
ENDLOOP.
```

13,506.74	SEK	
10,365.00	EUR	
527.91	GBP	
4,144.06	USD	