* **차례** –

[I. SELECT 구문 최적화 7](#_Toc411542512)

[1. SAP R/3 구조 이해하기 7](#_Toc411542513)

[2. SAP R/3 구조와 실행시간 7](#_Toc411542514)

[3. SELECT \* 개선 8](#_Toc411542515)

[4. 데이터 건수 확인 요령 8](#_Toc411542516)

[5. CHECK 구문 개선 9](#_Toc411542517)

[6. ORDER BY 구문 개선 10](#_Toc411542518)

[7. APPEND 구문 개선 10](#_Toc411542519)

[8. VIEW를 이용한 Nested SELECT 개선 11](#_Toc411542520)

[9. SELECT FOR ALL ENTRIES를 이용한 OUTER JOIN 12](#_Toc411542521)

[10. SQL 함수 사용하기 12](#_Toc411542522)

[11. WHERE 없는 SELECT 구문은 피하라. 13](#_Toc411542523)

[II. DML 구문 최적화 14](#_Toc411542524)

[1. 효과적인 UPDATE 기법 14](#_Toc411542525)

[2. 효과적인 INSERT 기법 14](#_Toc411542526)

[3. 효과적인 DELETE 기법 15](#_Toc411542527)

[III. 효과적인 Internal Table 작업 17](#_Toc411542528)

[1. Nested LOOP 처리 17](#_Toc411542529)

[2. 효과적인 LOOP 처리 17](#_Toc411542530)

[3. 내부 테이블 간 데이터 복사 18](#_Toc411542531)

[4. 내부 테이블의 데이터 APPEND 18](#_Toc411542532)

[5. 내부 테이블의 데이터 삭제 18](#_Toc411542533)

[6. 내부 테이블의 중복 데이터 삭제 19](#_Toc411542534)

[7. 내부 테이블에서 특정 필드의 값을 바꾸기 19](#_Toc411542535)

[8. 내부 테이블 데이터 개수 알아내기 19](#_Toc411542536)

[IV. 기타 20](#_Toc411542537)

[1. SYST 20](#_Toc411542538)

[2. 색상 색깔 COLOR 20](#_Toc411542539)

[3. lock mode 20](#_Toc411542540)

[3.1. E : 20](#_Toc411542541)

[3.2. S 20](#_Toc411542542)

[3.3. X 21](#_Toc411542543)

[4. Change Level 21](#_Toc411542544)

[4.1. 관련 T-code 21](#_Toc411542545)

[5. 도메인 값 얻어오기 21](#_Toc411542546)

[6. 통화키 얻어오기 21](#_Toc411542547)

[7. 통화의 소수점 자리수 얻어오기 및 금액에 적용 21](#_Toc411542548)

[8. 내부 테이블을 파일로 다운 받고 파일 열기. 22](#_Toc411542549)

[9. EXIT 22](#_Toc411542550)

[9.1. BADI 구현 순서 22](#_Toc411542551)

[9.2. 관련 BADI 알아내는 법 22](#_Toc411542552)

[9.3. SAPMF02K 23](#_Toc411542553)

[9.4. MEREQ001 23](#_Toc411542554)

[9.5. MM06E005 23](#_Toc411542555)

[10. Table Control 23](#_Toc411542556)

[10.1. Table Control을 화면 자체에서 조회가 되지 않게 하는 법. 23](#_Toc411542557)

[ENDLOOP. 23](#_Toc411542558)

[11. 문서에 링크 추가하기 23](#_Toc411542559)

[12. SAPGui 세션 수 늘리기 23](#_Toc411542560)

[13. Table 타입으로 내부 Table 처리 23](#_Toc411542561)

[V. Data Type 24](#_Toc411542562)

[1. Dict. data type ABAP data type 24](#_Toc411542563)

[2. 수량, 금액 필드 별 허용 값 24](#_Toc411542564)

[3. RFC Destination 25](#_Toc411542565)

[VI. T-Code 26](#_Toc411542566)

[1. 관리 26](#_Toc411542567)

[1.1. SCC1 26](#_Toc411542568)

[1.2. DB05 테이블 필드의 분산도 26](#_Toc411542569)

[1.3. FIBF: BTE (Business Transaction events) 26](#_Toc411542570)

[1.4. GRAL: 그래프 예제 26](#_Toc411542571)

[1.5. OAER: ALV Grid 용 그림 파일 26](#_Toc411542572)

[1.6. RZ11: 파라미터 관리 26](#_Toc411542573)

[1.7. RZ12: RFC 서버그룹 관리 26](#_Toc411542574)

[1.8. SE78: 그림파일 업로드(관련 SMW0, OAER) 26](#_Toc411542575)

[1.9. SMW0: 웹 관련 그림, 파일 업로드 26](#_Toc411542576)

[1.10. SPAD: 프린터 설정 26](#_Toc411542577)

[1.11. ST05 추적 26](#_Toc411542578)

[1.12. ST06: 서버 상태 확인 27](#_Toc411542579)

[1.13. ST12: ST05와 SE30의 기능을 합쳐놓은 것. 27](#_Toc411542580)

[1.14. STMS: CTS 관리자. 27](#_Toc411542581)

[1.15. SM36: 배치 작업 만들기 27](#_Toc411542582)

[1.16. SM37: 배치 작업 관리 27](#_Toc411542583)

[1.17. SM50: 실행 중인 프로그램 목록 보기. 27](#_Toc411542584)

[1.18. SMLG: 로드 발란싱 관리 27](#_Toc411542585)

[1.19. SU01: 사용자 등록정보(암호 변경 가능) 바꾸기. 27](#_Toc411542586)

[1.20. SU53: 권한 점검 27](#_Toc411542587)

[1.21. ST22: Dump 조회. 27](#_Toc411542588)

[1.22. SM66: 전체 시스템 프로세스 조회 27](#_Toc411542589)

[1.23. ST04: DB 성능 분석 27](#_Toc411542590)

[1.24. SM21: 시스템 로그 27](#_Toc411542591)

[1.25. OY04: 통화키 별 소수점 자리 수 지정. 27](#_Toc411542592)

[1.26. SAAB: 체크포인트 생성 27](#_Toc411542593)

[2. ITS 관리 27](#_Toc411542594)

[2.1. SICF: ITS, BSP 서비스 관리 27](#_Toc411542595)

[2.2. SITSPMON: 내부 ITS 상태 27](#_Toc411542596)

[2.3. ST11: ITS 덤프 보기. 27](#_Toc411542597)

[3. 개발 27](#_Toc411542598)

[3.1. LSMW: Legacy System Migration Workbench 27](#_Toc411542599)

[3.2. SCDO: 변경문서 관련 28](#_Toc411542600)

[3.3. SE63 : 언어 변경. 28](#_Toc411542601)

[3.4. SE75: 텍스트 오브젝트 설정 28](#_Toc411542602)

[3.5. Search\_sap\_menu: 메뉴 찾기. 28](#_Toc411542603)

[3.6. SQ01 : ABAP QUERY. 28](#_Toc411542604)

[3.7. AL11: 서버 폴더 목록 28](#_Toc411542605)

[3.8. SE84: Repository Info System. 28](#_Toc411542606)

[3.9. SE93: T-CODE 관리. 28](#_Toc411542607)

[3.10. SMARTFORMS – 스마트 폼, 스타일 편집. 28](#_Toc411542608)

[3.11. SE78 – 그림 등록. 28](#_Toc411542609)

[3.12. SWO2 비즈니스 오브젝트 관리 28](#_Toc411542610)

[4. 예제 프로그램 28](#_Toc411542611)

[4.1. DWDM: 예제 28](#_Toc411542612)

[4.2. AbapDocu: ABAP 프로그래밍 예제. 28](#_Toc411542613)

[4.3. BIBS: 화면요소예제 28](#_Toc411542614)

[5. 성능 확인 28](#_Toc411542615)

[5.1. SE30 : 프로그램, 함수 성능 검사. 28](#_Toc411542616)

[5.2. ST05 : 시스템 – 유틸리티 – 성능 추적. 28](#_Toc411542617)

[6. 이멜 전송 확인 28](#_Toc411542618)

[6.1. SOST : 이멜 전송 확인. 28](#_Toc411542619)

[7. 문서 관련 28](#_Toc411542620)

[7.1. CV01N : 문서 생성 28](#_Toc411542621)

[7.2. CV02N : 문서 조회 28](#_Toc411542622)

[7.3. CV03N : 문서 변경 28](#_Toc411542623)

[8. FI 모듈 28](#_Toc411542624)

[8.1. 전표관련 28](#_Toc411542625)

[VII. 프로그램 29](#_Toc411542626)

[1. 29](#_Toc411542627)

[1.1. RSTRANSP: 변형 이관 프로그램. 29](#_Toc411542628)

[1.2. RPR\_ABAP\_SOURCE\_SCAN: 소스 찾기. 29](#_Toc411542629)

[1.3. BALVBUFDEL: ALV 버퍼 초기화. 29](#_Toc411542630)

[VIII. 함수 30](#_Toc411542631)

[1. 메시지 관련 30](#_Toc411542632)

[1.1. MESSAGE\_TEXT\_BUILD 30](#_Toc411542633)

[1.2. SCPR\_SV\_SHOW\_MESSAGE\_LIST: 30](#_Toc411542634)

[1.3. TH\_POPUP 30](#_Toc411542635)

[2. 날짜 관련 30](#_Toc411542636)

[2.1. RP\_CALC\_DATE\_IN\_INTERVAL 30](#_Toc411542637)

[2.2. RP\_LAST\_DAY\_OF\_MONTHS 30](#_Toc411542638)

[2.3. DATE\_CONVERT\_TO\_FACTORYDATE 30](#_Toc411542639)

[3. 문자열 관련 30](#_Toc411542640)

[3.1. RKD\_WORD\_WRAP 30](#_Toc411542641)

[4. 변경문서 30](#_Toc411542642)

[4.1. CHANGEDOCUMENT\_READ\_HEADERS 30](#_Toc411542643)

[5. 숫자 관련 30](#_Toc411542644)

[5.1. 소수점이 포함된 문자를 숫자로 바꿔주기. 30](#_Toc411542645)

[5.2. 음수일 때 – 앞에 붙이기 (1,000,000 - 🡪 -1,000,000) 30](#_Toc411542646)

[5.3. 123,456.12 이런 숫자를 숫자인지 확인하는 함수. 30](#_Toc411542647)

[6. 배치작업 30](#_Toc411542648)

[6.1. 배치작업을 걸어주는 함수 30](#_Toc411542649)

[SUBST\_SCHEDULE\_BATCHJOB 30](#_Toc411542650)

[7. 구매모듈 내 자재, 업체 정보 조회 30](#_Toc411542651)

[8. 기타 31](#_Toc411542652)

[8.1. ABAP4\_CALL\_TRANSACTION 31](#_Toc411542653)

[8.2. 권한 없는 T-Code 실행 31](#_Toc411542654)

[C160\_TRANSACTION\_CALL 31](#_Toc411542655)

[8.3. 새 창을 열어 Transaction 코드 실행하기 31](#_Toc411542656)

[NAVIGATION\_EXECUTE\_OBJECT 31](#_Toc411542657)

[OBJECT\_NAME “ T-code 31](#_Toc411542658)

[8.4. 함수의 파라미터 목록 얻어오기 31](#_Toc411542659)

[RFC\_GET\_FUNCTION\_INTERFACE 31](#_Toc411542660)

[8.5. 인터넷 호출하기 31](#_Toc411542661)

[8.6. GFW\_PRES\_SHOW\_MULT: 그래프 출력(Business Graphic) 31](#_Toc411542662)

[GFW\_DEMO\_PRES, GFW\_DEMO\_PRES1, 31](#_Toc411542663)

[8.7. THUSRINFO 31](#_Toc411542664)

[8.8. SAPGUI\_PROGRESS\_INDICATOR 31](#_Toc411542665)

[8.9. TERM\_CONTROL\_EDIT 31](#_Toc411542666)

[8.10. SUSR\_PASSWORD\_CHANGE\_DIALOG 31](#_Toc411542667)

[8.11. TH\_DELETE\_USER 32](#_Toc411542668)

[8.12. DSYS\_SHOW\_FOR\_F1HELP 32](#_Toc411542669)

[8.13. SYSTEM\_CALLSTACK 32](#_Toc411542670)

[8.14. UA\_SE16N\_START 32](#_Toc411542671)

[8.15. VIEW\_AUTHORITY\_CHECK 32](#_Toc411542672)

[DATA : lv\_temp(30). 32](#_Toc411542673)

[SELECT SINGLE tname FROM zcdp\_ui\_table 32](#_Toc411542674)

[RAISE NO\_AUTHORITY. 32](#_Toc411542675)

[ENDIF. 32](#_Toc411542676)

[ENDIF. 32](#_Toc411542677)

[ENDENHANCEMENT. 32](#_Toc411542678)

[9. ALV 33](#_Toc411542679)

[9.1. 간단하게 ALV 팝업 띄우기 33](#_Toc411542680)

[REUSE\_ALV\_POPUP\_TO\_SELECT 33](#_Toc411542681)

[CALL FUNCTION 'REUSE\_ALV\_POPUP\_TO\_SELECT' 33](#_Toc411542682)

[IX. Web 34](#_Toc411542683)

[1. Abap 문법 34](#_Toc411542684)

[X. 테이블 35](#_Toc411542685)

[1. 개발 35](#_Toc411542686)

[1.1. E070 리퀘스트 상태 35](#_Toc411542687)

[1.2. TLOCK: 현재 개발 중 오브젝트. 35](#_Toc411542688)

[1.3. DD\* : 딕셔너리 구조 35](#_Toc411542689)

[1.4. TRMAC: 글로벌 매크로 설정. 35](#_Toc411542690)

[1.5. V\_OP: 배치작업 관련 35](#_Toc411542691)

[1.6. TPARA: 파라미터 ID 35](#_Toc411542692)

[1.7. TSTCP : PARAM 필드에 유지보수 뷰 이름 \* 으로 검색 35](#_Toc411542693)

[1.8. RFCDES: RFC 목적지 35](#_Toc411542694)

[2. 마스터 35](#_Toc411542695)

[2.1. T527X: 조직 마스터 35](#_Toc411542696)

[3. Long Text 35](#_Toc411542697)

[3.1. STXH: 테스트 헤더 35](#_Toc411542698)

[3.2. STXL: 텍스트 라인(SAP Script 형태) 35](#_Toc411542699)

[3.3. STXB: 텍스트 라인(기타 형태) 35](#_Toc411542700)

[4. 변경문서 35](#_Toc411542701)

[4.1. CDHDR: 변경문서 헤더 35](#_Toc411542702)

[4.2. CDPOS: 변경문서 품목 35](#_Toc411542703)

[XI. EXIT 36](#_Toc411542704)

[1.1. SUSR0001: 로그온 후 처리 됨. 36](#_Toc411542705)

[ZXUSRU01 36](#_Toc411542706)

[1.2. FMAVC\_ENTRY\_FILTER 36](#_Toc411542707)

[XII. 모듈 관련 37](#_Toc411542708)

[1. FI 37](#_Toc411542709)

[1.1. 전표유형 37](#_Toc411542710)

[1.2. 전기키 37](#_Toc411542711)

[XIII. 기타 38](#_Toc411542712)

[1. VB등에서 RFC 함수 호출하기 위해 필요한 OCX 38](#_Toc411542713)

# SELECT 구문 최적화

## SAP R/3 구조 이해하기

* R/3 구조를 이해하기 위해 아래와 같이 VBAK 테이블에서 데이터를 1000건 SELECT 하는 프로그램을 작성해 보자.
* 읽어올 정보는 다음과 같다.

- VBAK-VBELN: 오더 번호

- VBAK-AUART: 오더 종류

- VBAK-BNAME: 오더 이름

- VBAK-KUNNR: Sold to party

* 일반적으로 아래와 같은 방식으로 SELECT 문장을 작성할 것이다.

|  |
| --- |
| REPORT ZSAMPLE1.TABLES : vbak.SELECT \* FROM vbak.IF sy-dbcnt > 1000. EXIT. ENDIF.WRITE: / vbak-vbeln, …ENDSELECT. |

* 위 구문은 VBAK 테이블의 모든 칼럼을 한 건씩 읽어 가면서, WRITE 하는 구조로 되어 있으며, SY-DBCNT를 확인하여 1000건이 넘으면 EXIT하도록 되어 있다.
* 그럼 다음 단계에서 왜 위 문장이 비효율적인지 알아보자.

## SAP R/3 구조와 실행시간

* R/3 는 아래 그림과 같은 구조로 이루어져 있다.

Presentation Server

Application Server

Database

Server

Presentation Server

Application Server

* R/3 는 위의 그림과 같이 3 계층 구조로 되어 있다.
즉, 사용자가 시스템에 특정 데이터에 대한 요청을 하면, Application Server가 중간에서 사용자의 요청을 가공한 후, DB Server에 데이터에 대한 요청을 보내고, DB Server로부터 데이터를 받아, 다시 가공하여 사용자에게 보내는 방식으로 처리한다.
* 따라서, 앞과 같이 SELECT \* 하게 되면, 문제에서 제시한 네 개 필드 외에 해당 레코드의 모든 필드를 가져오게 된다. 즉, VBAK 테이블의 한 레코드 크기가 대략 575바이트인데, 1000건을 가져와야 하므로 총 575,000 바이트를 DB Server에서 Application Server로 전송해야 하며, 한번 전송할 때의 패킷 크기가 32,000 바이트인 경우(시스템에 따라 다를 수 있음), 총 20여 회의 네트워크 트래픽이 발생하게 된다. (575,000 / 32,000)
* 다음 단계에서 앞 문제의 해답과 이유를 알아보자.

## SELECT \* 개선

* SELECT \* 형태의 문장을 COLUMN Selection 이용하여 아래와 같이 개선할 수 있다.

|  |
| --- |
| SELECT vbeln auart bname kunnr INTO (vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr) FROM vbak WRITE : / vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr.ENDSELECT. |

* 위와 같이 사용하려는 네 개의 Column만을 가져오도록 할 경우 네트워크 트래픽은 다음과 같다.
* 즉, 네 개 Column 크기의 합이 59 바이트이므로 1000건을 읽는다 해도 59,000 바이트의 데이터만 DB Server에서 Application Server로 전송되면 된다. 패킷의 크기가 역시 32,000 이라면, 이 경우에는 두 번의 트래픽만 필요하게 된다.
* BSEG, VBFA 같은 SAP Cluster Table에 대해서는 Column Selection 을 할 수 없다(Column Selection을 하도록 작성해도 내부적으로는 전체 Column을 읽는다.). 이러한 테이블을 Column Selection으로 읽고 싶은 경우에는, 시스템을 내리고, 테이블 형태를 Cluster Table에서 Transparent Table로 바꾼다.

## 데이터 건수 확인 요령

* 아래 방법은 SY-DBCNT를 이용한 데이터 건수 확인 문장을 UP TO n ROWS 구문을 이용하여 개선하는 예를 보여준다.

|  |
| --- |
| \*-- 개 선 전SELECT vbeln auart bname kunnr INTO (vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr) FROM vbak. **IF SY-DBCNT > 1000. EXIT. ENDIF.** WRITE : / vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr.ENDSELECT.\*----------------------------------------------------------------------\*\*-- 개 선 후SELECT vbeln auart bname kunnr INTO (vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr)  FROM vbak **UP TO 1000 ROWS.** WRITE : / vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr.ENDSELECT. |

* 위의 예에서 첫 번째 문장은 데이터 1000건을 건건이 DB Server에서 Application Server로 전송하는 반면, 두 번째 문장은 DB Server에서 1000건을 읽은 후 일괄적으로 Application Server로 전송하는 방식으로 동작한다. 따라서 두번째 문장이 훨씬 좋은 성능을 나타낸다.
* WHERE 조건 없이 데이터 한 건만을 가져오는 경우라면, 차이가 더욱 두드러진다. 한 건을 가져올 경우 전자의 경우 150,000ms, 후자의 경우 1,500ms 시간이 걸린다.

## CHECK 구문 개선

* 아래 예는 CHECK 문장에 들어가는 조건을 SELECT 문장에 포함하여 개선하는 방법이다.

|  |
| --- |
| PARAMETERS : param1.DATA : BEGIN OF search\_string, FIRST(9) VALUE ‘\_\_\_\_\_\_\_\_\_’, PARAM, END OF search\_string.\*-- 개 선 전SELECT vbeln auart bname kunnr INTO (vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr) FROM vbak. **CHECK vbak-vbeln+9(1) = param1.** WRITE : / vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr.ENDSELECT.\*----------------------------------------------------------------------\*\*-- 개 선 후search\_string-param = param1.SELECT vbeln auart bname kunnr INTO (vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr) FROM vbak **WHERE vbeln LIKE search\_string.** WRITE : / vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr.ENDSELECT. |

* Vbak 테이블에서 57,000건을 가져오는 것을 비교한 경우

SELECT & CHECK 🡪 27,958,000ms

SELECT WITH WHERE condition 🡪 3,065,000ms

* 가능하다면 어떠한 경우라도, CHECK 조건은 WHERE 절 안에 들어와야 하며 이것은 SELECT SINGLE에도 적용된다.

 SELECT SINGLE vbeln auart … INTO (vbak-vbeln, …) FROM vbak

 WHERE vbeln = ‘0090000090’ AND bname = ‘smith’.

* 위 구문에서 vbak 테이블의 키 필드는 vbeln이지만, bname을 CHECK로 빼지 않고, WHERE 절에 포함시킴으로써, DB Server와 Application Server 간에 쓸데없는 네트워크 트래픽을 줄일 수 있다.
* 그러나, CHECK 절이 매우 까다롭고 복잡한 경우에는 오히려 DB Server에서 일단 데이터를 받아오는 것이 빠를 수도 있다. 이것은 SQL Trace를 이용하여 실제로 어떤 것이 빠른지 조사해보고 결정하여야 한다.

## ORDER BY 구문 개선

* 아래 예는 SELECT 구문에서 사용되는 ORDER BY를 Internal Sort를 이용하여 개선하는 방법을 보여준다.

|  |
| --- |
| \*-- 개 선 전SELECT vbeln auart bname kunnr INTO (vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr) FROM vbak WHERE bname IN so\_bname **ORDER BY bname DESCENDING**.ENDSELECT.\*----------------------------------------------------------------------\*\*-- 개 선 후SELECT vbeln auart bname kunnr INTO TABLE it\_vbak FROM vbak WHERE bname IN so\_bname.**SORT it\_vbak BY bname DESCENDING.** |

* 많은 양의 DB를 DB Server에서 정렬하는 것은 그 시스템을 사용중인 모든 사용자에게 영향을 미친다. 하지만 ABAP/4 프로그램 내에서 처리하게 되면 해당 Application Server에만 영향을 미친다.
* 하지만 SELECT 구문 내에서 ORDER BY를 이용하여 정렬하고자 하는 필드에 INDEX가 적절하게 구성되어 있다면, DB Server에 해롭지 않다.

## APPEND 구문 개선

* 아래 예는 APPEND 구문을 이용하여 내부 테이블에 건건이 데이터를 추가하는 것을 INTO TABLE을 사용하여 개선하는 방법을 보여준다.

|  |
| --- |
| \*-- 개 선 전SELECT vbeln auart bname kunnr INTO (it\_vbak-vbeln, it\_vbak-auart, it\_vbak-bname, it\_vbak-kunnr) FROM vbak WHERE bname IN so\_bname. **APPEND it\_vbak.** **“ 추가 작업.**ENDSELECT.\*----------------------------------------------------------------------\*\*-- 개 선 후SELECT vbeln auart bname kunnr **INTO TABLE it\_vbak** FROM vbak WHERE bname IN so\_bname.LOOP AT it\_vbak. “ 추가 작업.ENDLOOP. |

* 내부 테이블의 내용을 한 번에 전송하는 것이 더 효과적이다.
* 데이터를 내부 테이블로 한 번에 읽어 들이는 방법의 또 다른 효과는 처리시간이 매우 짧다는 것이다.
* SELECT .. ENDSELECT. 내에서 많은 처리를 할수록 ORAO155 Snapshot too old(rollback segment too small) 오류를 만날 확률이 높다.
* 수행시간이 긴 SELECT 문장은 동일한 작업 대상 테이블에 대한 INSERT/UPDATE 문장과 I/O Contention을 일으킬 확률이 높으며, 이럴 경우, DB Server가 반드시 read consistency를 보장한다고 단언하기 어렵다.

## VIEW를 이용한 Nested SELECT 개선

* 아래 예는 여러 개의 테이블에서 데이터를 가져올 때, view를 이용하여 NESTED SELECT를 개선하는 방법을 보여준다.

|  |
| --- |
| \*-- 개 선 전**SELECT** vbeln auart bname kunnr INTO (vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr) FROM vbak WHERE vbeln IN so\_vbeln.  **SELECT** posnr matnr kwmeng meins INTO … FROM vbap WHERE vbeln = vbak-vbeln. WRITE : / vbak-vbeln, vbak-auart, vbak-bname, vbak-kunnr. **ENDSELECT**.**ENDSELECT**.\*----------------------------------------------------------------------\*\*-- 개 선 후SELECT \* FROM **zv\_vbak “ VIEW** WHERE vbeln IN so\_vbeln. WRITE : / …ENDSELECT. |

* VBAK : 716건, VBAP 2779건의 데이터를 대상으로 테스트 한 결과는 아래와 같다.
	+ 개선 전: 3,892,184 microseconds(약 4초)
	+ 개선 후: 981,385 microseconds(약 1초)
* Nested SELECT 대신에 VIEW를 이용하여 두 개의 테이블을 JOIN한 다음, VIEW에서 데이터를 읽어오는 것이 훨씬 수행 속도가 빠르다.
* 그러나, DB VIEW는 버퍼를 경유하지 않고 바로 DB에서 데이터를 읽어 오기 때문에 두 개의 테이블 중 하나가 버퍼링이 되고 있는 경우, 버퍼링의 장점을 살리지 못하여, 늦어지는 경우가 아주 가끔 발생할 수도 있다. 이런 경우는 SQL Trace를 이용하여 실제로 수행시간을 측정하고 결과가 좋은 것을 사용하도록 한다.
* DB Dictionary 의 DB VIEW는 항상 INNER JOIN이다.
* VIEW생성시 MANDT열을 가져오도록 정의한 테이블을 가장 먼저 읽는다.

## SELECT FOR ALL ENTRIES를 이용한 OUTER JOIN

* 아래는 SELECT FOR ALL ENTRIES를 이용하여 ABAP에서 OUTER JOIN을 구현한 예이다.

|  |
| --- |
| \*-- 개선 전.SELECT vbeln auart … INTO … FROM vbak WHERE ….SELECT posnr matnr … INTO … FROM vbap WHERE vbeln = vbak-vbeln.… ENDSELECT.ENDSELECT.\*----------------------------------------------------------------------\*\*-- 개선 후.SELECT vbeln auart… INTO TABLE it\_vbak FROM vbakWHERE vbeln IN so\_vbeln.SELECT vbeln posnr matnr … INTO TABLE it\_vbap FROM vbap **FOR ALL ENTRIES IN it\_vbak** WHERE vbeln = it\_vbak-vbeln. “ vbeln과 it\_vbak-vbeln은 같은 형과 길이여야 한다.LOOP AT it\_vbak.READ TABEL it\_vbap WITH KEY vbeln = it\_vbak-vbeln BINARY SEARCH TRANSPORTING NO FIELDS.LOOP AT it\_vbap FROM sy-tabix. …ENDLOOP.ENDLOOP.  |

* OUTER JOIN을 할 경우 위처럼 FOR ALL ENTRIES IN 구문을 이용하여 SELECT한 후, 내부 테이블에서 처리하도록 한다.

## SQL 함수 사용하기

* SELECT구문에서 LOOP를 돌면서 처리하는 형태의 합, 평균을 구하는 방법을 SQL 함수를 이용하여 개선하는 예이다.

|  |
| --- |
| DATA : BEGIN OF vbqty OCCURS 0. matnr LIKE vbap-matnr, kwmeng LIKE vbap-kwment, meins LIKE vbap-meins.\*-- 개선 전.SELECT matnr kwmeng meins INTO vbqty FROM vbap WHERE … . COLLECT vbqty.ENDSELECT.\*----------------------------------------------------------------------\*\*-- 개선 후.SELECT matnr **SUM**( kwmeng ) meins INTO TABLE vbqty FROM vbap WHERE …  GROUP BY matnr meins. |

* 10,000건의 데이터를 처리하였을 경우, 아래와 같은 성능 차이가 나왔다.
	+ 개선 전: 2,370,000 microseconds
	+ 개선 후: 1,574,000 microseconds
* GROUP BY 절은 Pool Table이나 Cluster Table에는 사용될 수 없다(BSEG, VBFA).

## WHERE 없는 SELECT 구문은 피하라.

* 아래와 같이 WHERE 조건 없이 SELECT 구문을 사용하는 것은 매우 위험하다.

|  |
| --- |
| SELECT col1 col2 … INTO … FROM vbak. PERFORM cal\_rtn.ENDSELECT. |

* 특히 다음과 같이 빠른 속도로 크기가 증가하는 테이블에는 절대로 사용하지 말 것
* BKPF, BSEG, COBK, COEP, LIPK, LIPS, MKPF, MSEG, VBAK, VBAP, VBPA, VBFA
* 테이블의 모든 데이터를 읽어 작업을 수행하려 한다면 다음과 같이 EXEC SQL 구문을 이용한 native sql을 구사하여 Application Server를 Bypass 하도록 한다. 그렇지 않으면 Application Server가 Where 절에 MANDT 조건을 붙여, Full Table Scan이 되지 않고 인덱스를 이용하게 되어 최악의 결과를 초래하게 된다.

|  |
| --- |
| EXEC SQL.SELECT /\* rule \*/ Vbeln INTO : ivbak-vbelnFROM vbakWHERE mandt || ‘’ =: sy-mandtENDEXEC. |

# DML 구문 최적화

## 효과적인 UPDATE 기법

* SELECT 와 ENDSELECT 사이에서 순환을 하면서 UPDATE 하는 형태는 아래와 같이 WHERE 절을 이용하여 개선할 수 있다.

|  |
| --- |
| \*-- 개선 전.PARAMETES : pa\_vkbur LIKE zvbak-vkbur.SELECT-OPTIONS : so\_vbeln FOR zvbak-vbeln.SELECT \* FROM zvbak WHERE vbeln IN so\_vbeln. zvbak-vkbur = pa\_vkbur. UPDATE zvbak.EDNSELECT.\*----------------------------------------------------------------------\*\*-- 개선 후.UPDATE zvbak SET vkbur = pa\_vkbur WHERE vbeln IN so\_vbeln.IF sy-subrc = 0. COMMIT WORK.ELSE. ROLLBACK WORK.ENDIF. |

* zvbak 1000건을 대상으로 한 결과는 다음과 같다.

SELECT .. UPDATE .. ENDSELECT. 🡪 1.8초

UPDATE .. SET 🡪 0.3초

* 위의 예제에서 COMMIT WORK 또는 ROLLBACK WORK 구문은 논리적인 작업 단위가 끝나면 반드시 기술해 주는 것을 잊지 말자. COMMIT WORK나 ROLLBACK WORK를 빠뜨리면 DB Server가 수정 정보를 계속 rollback segment에 가지고 있어야 하기 때문에 System에 overload로 계속 남아있게 된다.

## 효과적인 INSERT 기법

* 데이터를 테이블에 INSERT 하고자 할 경우, LOOP를 돌면서 처리하는 것보다 아래의 예처럼 한꺼번에 INSERT 하면 효과적이다.

|  |
| --- |
| \*-- 개선 전.LOOP at it. MOVE it TO z100. INSERT z100.ENDLOOP.\*----------------------------------------------------------------------\*\*-- 개선 후.INSERT z100 FROM TABLE it. |

* 아래 결과는 건수 당 테스트 결과이다.
* 1,000건 입력 시.

한 건씩 : 2.58초

한꺼번에 : 0.72초

* 5,000건 입력 시.

한 건씩 : 9.66초

한꺼번에 : 1.59초

* 10,000건 입력 시.

한 건씩 : 20.00초

한꺼번에 : 2.47초

## 효과적인 DELETE 기법

* SELECT 와 ENDSELECT 사이에서 순환을 하면서 DELETE 하는 형태는 아래와 같이 WHERE 절을 이용하여 개선할 수 있다.

|  |
| --- |
| \*-- 개선 전.SELECT-OPTIONS : so\_vbeln FOR zvbak-vbeln.SELECT \* FROM zvbak WHERE vbeln IN so\_vbeln. DELETE zvbak.EDNSELECT.\*----------------------------------------------------------------------\*\*-- 개선 후.DELETE FROM zvbak WHERE vbeln IN so\_vbeln.IF sy-subrc = 0. COMMIT WORK.ELSE. ROLLBACK WORK.ENDIF. |

* zvbak 1000건을 대상으로 한 결과는 다음과 같다.

SELECT .. DELETE .. ENDSELECT. 🡪 3.0초

DELETE FROM 🡪 1.2초

* 개선 전 코딩은 거의 최악의 코딩이다. 개선 전 경우 DB Server는 대상 데이터를 찾아서 모두 Application Server로 보낸다. 데이터를 받은 Application Server는 아무 일도 하지 않고, 받은 데이터를 다시 DB Server로 보내고, DB Server에서는 이 데이터를 삭제한다.
* 그러나 개선 후의 경우는 데이터의 발췌 및 삭제 작업이 DB Server에서만 발생하게 되므로, Application Server와 DB Server사이에 데이터 전송 작업이 전혀 일어나지 않는다.

# 효과적인 Internal Table 작업

## Nested LOOP 처리

* Nested LOOP는 BINARY SEARCH를 이용하여 아래와 같이 개선할 수 있다.

|  |
| --- |
| DATA : ivbak LIKE VBAK OCCURS 0,Ivbap LIKE VBAP OCCURS 0.SORT : ivbaK BY vebln, ivbap BY vbeln.\*-- 개선 전.LOOP AT ivbak. LOOP AT ivbap WHERE vbeln = ivbak-vbeln. WRITE : / … ENDLOOP.ENDLOOP.\*-- 개선 후.LOOP AT ivbak. “ Work Area에 아무 값도 넘기지 않음. Sy-subrc, sy-tabix를 설정하기 위함. READ TABLE ivbap WITH KEY vbeln = ivbak-vbeln BINARY SEARCH TRASPORTING NO FIELDS. LOOP AT ivbap FROM sy-tabix. IF ivbap-vbeln <> ivbak-vbeln. EXIT. ENDIF. WRITE : /… ENDLOOP.ENDLOOP. |

* Ivbak : 5,000 ivbap : 20,000 일 때.

Nested LOOP with WHERE : 5분.

LOOP/READ BINARY SEARCH/LOOP : 15초.

## 효과적인 LOOP 처리

* 특정 조건을 만족하는 데이터가 몇 건인지를 알고 싶은 경우에는 다음과 같이 한다.

|  |
| --- |
| LOOP AT it1 **TRANSPORTING NO FIELDS** WHERE bname = ‘smith’. ADD 1 TO counter.ENDLOOP. |

* 특정 조건을 만족시키는 데이터가 있는지 알고 싶다면 아래와 같이 하라.

|  |
| --- |
| LOOP AT it1 **TRANSPORTING NO FIELDS** WHERE vbeln > ‘003000000100’. EXIT.ENDLOOP. |

## 내부 테이블 간 데이터 복사

* 동일한 구조를 가지는 두 개의 내부 테이블 간에 서로 데이터를 복사하고자 할 경우 아래와 같이 한다.

|  |
| --- |
| DATA : it1 LIKE eban OCCURS 100 WITH HEADER LINE, It2 LIKE eban OCCURS 100 WITH HEADER LINE.\*-- 개선 전.REFRESH it2.LOOP AT it1. MOVE it1 TO it2. APPEND it2.ENDLOOP.\*-- 개선 후.It2[] = it1[]. |

## 내부 테이블의 데이터 APPEND

* 동일한 구조를 가지는 두 개의 내부 테이블간에 서로 데이터를 APPEND 하고자 할 경우에는 아래와 같이 개선한다.

|  |
| --- |
| DATA : it1 LIKE eban OCCURS 100 WITH HEADER LINE, It2 LIKE eban OCCURS 100 WITH HEADER LINE.\*-- 개선 전.LOOP AT it1. MOVE it1 TO it2. APPEND it2.ENDLOOP.\*-- 개선 후.APPEND LINES OF it1 TO it2. |

## 내부 테이블의 데이터 삭제

* 여러 건의 데이터를 내부 테이블에서 삭제하고자 할 경우 LOOP를 돌지 말고 아래와 같이 한다.

|  |
| --- |
| \*-- 개선 전.LOOP AT ivbak WHERE bname = ‘smith’. DELETE ivbak.ENDLOOP.\*-- 개선 후.DELETE ivbak WHERE bname = ‘smith’. |

## 내부 테이블의 중복 데이터 삭제

|  |
| --- |
| DELETE ADJACENT DUPLICATES FROM itab [COMPARING field1 field2…]. |

* 위 방법을 사용하기 위해서는 비교조건 필드를 기준으로 정렬되어 있어야 한다. 비교조건이 없다면, 모든 필드로 정렬되어 있어야 한다.

## 내부 테이블에서 특정 필드의 값을 바꾸기

|  |
| --- |
| GT\_POLIST-MARK = ' '.MODIFY GT\_POLIST TRANSPORTING MARK WHERE MARK = 'X' . |

## 내부 테이블 데이터 개수 알아내기

|  |
| --- |
| **DESCRIBE TABLE itab LINES n.** |

# 기타

## SYST

SY-LISEL : 더블 클릭 시, 행의 내용을 가지고 있음

SY-LILLI : 더블 클릭 시, 행의 번호를 가지고 있음

SY-CUCOL : 현재 CURSOR가 있는 열의 번호

SY-CUROW : 현재 CURSOR가 있는 행의 번호

## 색상 색깔 COLOR

CXYZ 로 표시

X: 색상

Y: 1/0 Intensified on/off 강조

Z: 1/0 Inverse on/off 역

|  |  |  |
| --- | --- | --- |
| X | 색상 | 목적 |
| 1 | Gray-blue | 헤더 |
| 2 | Light gray | 리스트 |
| 3 | Yellow | Total |
| 4 | Blue-green | Key |
| 5 | Green | 양호한 값 |
| 6 | Red | 부정적인 값 |
| 7 | Orange | Control level |



## lock mode

### E :

Exclusive, Cumulative : 오직 한 명의 사용자에 대해서만 테이블 접근이 가능하다. 다른 Exclusive 또는 Shared 를 수용하지 않는다.

### S

 Shared : 여러 명의 사용자가 같은 시간에 같은 데이터를 읽을 수 있다. 하지만 한 사용자가 그 데이터를 변경하기 시작한다면 다른 사용자는 더 이상 그 데이터에 접근할 수 없다. 또한 다른 사용자에 의해 발생하는 Shared 는 수용하지만 Exclusive는 수용하지 않는다.

### X

Exclusive, not Cumulative : 하나 이상의 동일한 Transaction 내에서 lock을 요청할 수 있다. 그러나 Exclusive, not Cumulative lock은 주어진 트랜잭션 내에서 단 한 번만 락을 요청할 수 있으며 모든 락 요청을 수용하지는 않는다.

## Change Level

|  |  |
| --- | --- |
| SAP Standard Changing | 예: 수도관 파이프 |
| Customizing | Implemetaion Guide | 집 구조를 고려하여 수도 파이프를 배치 |
| Personalization | Transaction Variants | 각방의 용도에 맞게 수도 꼭지를 배치 |
| Modification | Modification AssistantUser Exits | 파이프의 중간을 잘라서 연결 |
| Enhancement | ABAP DictionaryCustomer ExitsBusiness Transaction EventsBusiness Add-Ins | 파이프와 파이프의 연결 부위에 파이프를 추가 |
| Development | ABAP Developer | 현재의 파이프로는 연결하기 힘든 부분을 새로운 파이프를 만들어서 연결. |

### 관련 T-code

#### SMOD: exit 오브젝트

#### CMOD: 프로젝트 관리

## 도메인 값 얻어오기

|  |
| --- |
| SELECT SINGLE ddtext FROM dd07v INTO p\_text “ CHAR 60 WHERE domname = p\_domain “ 도메인 이름 AND domvalue\_l = p\_value “ 도메인 값 AND ddlanguage = sy-langu. |

## 통화키 얻어오기

|  |
| --- |
| SELECT SINGLE ktext FROM tcurt INTO p\_waerstxt WHERE waers = p\_waers AND spras = sy-langu. |

## 통화의 소수점 자리수 얻어오기 및 금액에 적용

|  |
| --- |
| DATA : lv\_cd LIKE bapi1090\_1. CALL FUNCTION 'BAPI\_CURRENCY\_GETDECIMALS' EXPORTING currency = gs\_pohead-waers IMPORTING currency\_decimals = lv\_cd. lv\_cd-curdecimals = 2 - lv\_cd-curdecimals. DO lv\_cd-curdecimals TIMES. lv\_temp = lv\_temp \* 10. ENDDO. |

## 내부 테이블을 파일로 다운 받고 파일 열기.

|  |
| --- |
| call function '**DOWNLOAD**' exporting filename = filename filetype = filetype mode = mode importing act\_filename = act\_filename tables data\_tab = down\_itab.call function '**WS\_EXECUTE**' exporting program = act\_filename inform = '' exceptions prog\_not\_found. |

## EXIT

### BADI 구현 순서

1. Enhancement Spot 생성

2. BADI Definition 생성

3. Enhancement Implementation 생성

4. BADI Implementation 생성

### 관련 BADI 알아내는 법

클래스 CL\_EXITHANDLER의 get\_instance 에서 아래에 디버깅 걸고 프로그램 실행하면 exit\_name에 바디 이름 있음.

CALL METHOD cl\_exithandler=>get\_class\_name\_by\_interface

### SAPMF02K

구매처 마스터 EXIT.

### MEREQ001

구매요청 EXIT

### MM06E005

구매오더 EXIT

## Table Control

### Table Control을 화면 자체에서 조회가 되지 않게 하는 법.

DATA :wa TYPE cxtab\_column.

 LOOP AT tc0100\_1-cols INTO wa.

 wa-screen-invisible = 1.

 wa-invisible = 1.

 MODIFY tc0100\_1-cols FROM wa.

ENDLOOP.

## 문서에 링크 추가하기

트랜잭션 실행: <DS:TRAN.*t-code*>*보여줄 내용*</>

리포트 프로그램 실행: <DS:REPN.*리포트 프로그램이름*>*보여줄 내용*</>

다른 오브젝트 문서 호출: <DS:*id.object*>*보여줄 내용*</>

※ DOKTL 테이블의 id.object

개발 오브젝트 문서 내용을 가지고 있는 Table: DOKTL

관련 함수: DOCU\_READ

## SAPGui 세션 수 늘리기

T-Code RZ11 실행하여 rdisp/max\_alt\_modes 변경.

## Table 타입으로 내부 Table 처리

DATA gt\_item1 TYPE y0tttesti1\_av WITH HEADER LINE.

FORM USING ps\_item TYPE LINE OF y0tttesti1\_av.

FORM TABLES pt\_item TYPE y0tttesti1\_av.

# Data Type

## Dict. data type ABAP data type

* ACCP -> N(6)
* CHAR n -> C(n)
* CLNT -> C(3)
* CUKY -> C(5)
* CURR n, m, s -> P((n + 2) / 2) DECIMALS m [NO-SIGN]
* DEC n, m, s -> P((n + 2) / 2) DECIMALS m [NO-SIGN]
* DATS -> D
* FLTP -> F
* INT1 -> No corresponding type
* INT2 -> No corresponding type
* INT4 -> I
* LCHR n -> C(n)
* LRAW n -> X(n)
* LANG -> C(1)
* NUMC n -> N(n)
* PREC -> X(2)
* QUAN n, m, s -> P((n + 2) / 2) DECIMALS m [NO-SIGN]
* RAW n -> X(n)
* TIMS -> T
* UNIT n -> C(n)
* VARC n -> C(n)

## 수량, 금액 필드 별 허용 값

| 도메인길이 | 최대값 | 최대값 예 | 출력길이 |
| --- | --- | --- | --- |
| CURR11 | 100억 | 11,111,111,111 | 14 |
| CURR13 | 1조 | 1,111,111,111,111 | 17 |
| CURR15 | 100조 | 111,111,111,111,111 | 20 |
| CURR17 | 1경 | 11,111,111,111,111,111 | 22 |
| CURR19 | 100경 | 1,111,111,111,111,111,111 | 25 |
|  |  |  |  |
| QUAN11 | 천만 | 11,111,111.999 | 14 |
| QUAN13 | 십억 | 1,111,111,111.999 | 17 |
| QUAN15 | 천억 | 111,111,111,111.999 | 19 |
| QUAN17 | 십조 | 11,111,111,111,111.999 | 22 |
| QUAN19 | 천조 | 1,111,111,111,111,111.999 | 25 |

## RFC Destination

|  |
| --- |
| DATA : rfcdest LIKE rfcdisplay-rfcdest. |

# T-Code

## 관리

### SCC1

타 클라이언트의 CTS 가져오기

### DB05 테이블 필드의 분산도

INDEX는 분산도(Distinct values)가 높은 필드를 위주로 만드는 것이 좋다.

### FIBF: BTE (Business Transaction events)

### GRAL: 그래프 예제

### OAER: ALV Grid 용 그림 파일

JPG 파일도 가능

BDS\_BUSINESSDOCUMENT\_GET\_URL을 통해 접근.

### RZ11: 파라미터 관리

rdisp/max\_debug\_lazy\_time 디버거 화면 활성 시간

dbs/ora/array\_buf\_size Fetch Size

### RZ12: RFC 서버그룹 관리

Logon Group을 이용해서 병렬 처리를 할 수 있다.

RFC 호출 시 “DESTINATION IN GROUP ” 으로 주면 됨.

### SE78: 그림파일 업로드(관련 SMW0, OAER)

스마트 폼 또는 SAP 스크립트에서 사용하는 이미지 등록

BMP, TIFF 파일 256색까지만 지원

CL\_SSF\_XSF\_UTILITIES=>GET\_BDS\_GRAPHIC\_AS\_BMP 를 이용해서 가져옴.

### SMW0: 웹 관련 그림, 파일 업로드

WWW\_GET\_MIME\_OBJECT 함수 이용 접근.

엑셀 파일 등을 올려놓고 프로그램에서 해당 파일에 값을 채워서 내려 받기 기능 구현 가능.

### SPAD: 프린터 설정

외국어 출력 시 유용.

### ST05 추적

[필터로 추적] -> 프로그램 실행 -> [추적해제] -> [추적리스트]

### ST06: 서버 상태 확인

### ST12: ST05와 SE30의 기능을 합쳐놓은 것.

### STMS: CTS 관리자.

### SM36: 배치 작업 만들기

### SM37: 배치 작업 관리

### SM50: 실행 중인 프로그램 목록 보기.

* DIA: 하나의 화면을 처리하는 프로세스
* UPD: DB 변경을 처리하는 UPDATE
* UP2: 지연된 UPDATE를 처리하는 UPDATE 프로세스
* ENQ: SAP LOCK 을 잡거나 풀어주는 프로세스
* BGD: BATCH JOB을 처리하는 BACKGROUND 프로세스
* SPO: 프린트를 담당하는 Spool 프로세스.

### SMLG: 로드 발란싱 관리

### SU01: 사용자 등록정보(암호 변경 가능) 바꾸기.

### SU53: 권한 점검

### ST22: Dump 조회.

### SM66: 전체 시스템 프로세스 조회

### ST04: DB 성능 분석

### SM21: 시스템 로그

### OY04: 통화키 별 소수점 자리 수 지정.

### SAAB: 체크포인트 생성

## ITS 관리

### SICF: ITS, BSP 서비스 관리

### SITSPMON: 내부 ITS 상태

### ST11: ITS 덤프 보기.

## 개발

### LSMW: Legacy System Migration Workbench

외부 데이터를 올릴 때 편함.

### SCDO: 변경문서 관련

### SE63 : 언어 변경.

### [SE75: 텍스트 오브젝트 설정](#_Long_Text)

### Search\_sap\_menu: 메뉴 찾기.

### SQ01 : ABAP QUERY.

### AL11: 서버 폴더 목록

### SE84: Repository Info System.

### SE93: T-CODE 관리.

### SMARTFORMS – 스마트 폼, 스타일 편집.

### SE78 – 그림 등록.

### SWO2 비즈니스 오브젝트 관리

## 예제 프로그램

### DWDM: 예제

### AbapDocu: ABAP 프로그래밍 예제.

### BIBS: 화면요소예제

## 성능 확인

### SE30 : 프로그램, 함수 성능 검사.

### ST05 : 시스템 – 유틸리티 – 성능 추적.

## 이멜 전송 확인

### SOST : 이멜 전송 확인.

## 문서 관련

### CV01N : 문서 생성

### CV02N : 문서 조회

### CV03N : 문서 변경

## FI 모듈

### 전표관련

#### F-02: G/L 계정 전기

#### F-22: 고객 송장

#### F-27: 고객 대변메모

#### F-43: 구매처 송장

#### F-47: 구매처 대변메모

#### FB03: 전표 조회

# 프로그램

##

### RSTRANSP: 변형 이관 프로그램.

### RPR\_ABAP\_SOURCE\_SCAN: 소스 찾기.

### BALVBUFDEL: ALV 버퍼 초기화.

# 함수

## 메시지 관련

### MESSAGE\_TEXT\_BUILD

메시지 ID, 번호, Parameter를 가지고 긴 텍스트 만들어 주기

### SCPR\_SV\_SHOW\_MESSAGE\_LIST:

다수의 메시지를 화면에 출력

### TH\_POPUP

로그인 한 사용자에게 메시지 팝업 띄움

## 날짜 관련

### RP\_CALC\_DATE\_IN\_INTERVAL

현재일로부터 특정일 계산하기

### RP\_LAST\_DAY\_OF\_MONTHS

입력한 날의 달 마지막 날 구하기.

### DATE\_CONVERT\_TO\_FACTORYDATE

공장 달력 이용하여 유용한 업무 일자 가져오기.

## 문자열 관련

### RKD\_WORD\_WRAP

긴 문자열을 일정한 길이로 잘라주는 기능

## 변경문서

### CHANGEDOCUMENT\_READ\_HEADERS

## 숫자 관련

### 소수점이 포함된 문자를 숫자로 바꿔주기.

* CHAR\_FLTP\_CONVERSION

### 음수일 때 – 앞에 붙이기 (1,000,000 - 🡪 -1,000,000)

* CLOI\_PUT\_SIGN\_IN\_FRONT

### 123,456.12 이런 숫자를 숫자인지 확인하는 함수.

* CATS\_NUMERIC\_INPUT\_CHECK

## 배치작업

### 배치작업을 걸어주는 함수

SUBST\_SCHEDULE\_BATCHJOB

## 구매모듈 내 자재, 업체 정보 조회

MEEXT 함수 그룹 참고.

## 기타

### ABAP4\_CALL\_TRANSACTION

T-Code 실행.

### 권한 없는 T-Code 실행

C160\_TRANSACTION\_CALL

### 새 창을 열어 Transaction 코드 실행하기

NAVIGATION\_EXECUTE\_OBJECT

 OBJECT\_NAME “ T-code

 REPORTTYPE = ‘TR’

 NEW\_WINDOW = ‘X’

### 함수의 파라미터 목록 얻어오기

RFC\_GET\_FUNCTION\_INTERFACE

### 인터넷 호출하기

|  |
| --- |
| CALL FUNCTION 'CALL\_BROWSER' EXPORTING url = lv\_url window\_name = 'WORKFLOW' new\_window = 'X'. |

### GFW\_PRES\_SHOW\_MULT: 그래프 출력(Business Graphic)

관련 예제 프로그램:

GFW\_DEMO\_PRES, GFW\_DEMO\_PRES1,

DEMO\_GFW\_PRES\_SHOW, DEMO\_GFW\_PRES\_SHOW\_MULT,

GFW\_DEMO\_HIER1, GFW\_DEMO\_HIER2, GFW\_DEMO\_HIER3

### THUSRINFO

현재 접속 중인 사용자 목록

### SAPGUI\_PROGRESS\_INDICATOR

진행 보여주기

### TERM\_CONTROL\_EDIT

긴 문장을 입력 받을 수 있는 창 띄우기.

### SUSR\_PASSWORD\_CHANGE\_DIALOG

사용자 암호 변경 창 띄우기

### TH\_DELETE\_USER

사용자 로그오프

### DSYS\_SHOW\_FOR\_F1HELP

오브젝트 문서 보여주기.

### SYSTEM\_CALLSTACK

해당 시점까지 거쳐왔던 프로그램, 함수, 서브루틴 목록.

### UA\_SE16N\_START

Table 내용 변경.

### VIEW\_AUTHORITY\_CHECK

Table, View의 내용 조회를 막을 수 있음.

FUNCTION view\_authority\_check.

"$"$\SE:(1) Function Module VIEW\_AUTHORITY\_CHECK, Start

\*$\*$-Start: (1)---------------------------------------------------------------------------------$\*$\*

ENHANCEMENT 3 ZCDP\_SE12\_BLOCK. "active version

\* put the detail logic here.

 DATA : lv\_temp(30).

 SELECT SINGLE tname FROM zcdp\_ui\_table

 INTO lv\_temp

 WHERE tname = view\_name.

 IF sy-subrc = 0.

 SELECT SINGLE uname FROM agr\_users

 INTO lv\_temp

 WHERE agr\_name = 'ZADM\_TABLE\_DISPLAY'

 AND uname = sy-uname.

 IF sy-subrc NE 0.

 RAISE NO\_AUTHORITY.

 ENDIF.

 ENDIF.

ENDENHANCEMENT.

## ALV

### 간단하게 ALV 팝업 띄우기

REUSE\_ALV\_POPUP\_TO\_SELECT

CALL FUNCTION 'REUSE\_ALV\_POPUP\_TO\_SELECT'

 EXPORTING

 i\_title = lv\_title

 i\_selection = ' '

 i\_screen\_start\_column = 3

 i\_screen\_start\_line = 3

 i\_screen\_end\_column = 100

 i\_screen\_end\_line = 12

 i\_tabname = 'GT\_LABST'

 it\_fieldcat = lt\_fieldcat

 is\_private = ls\_private

 i\_callback\_program = sy-cprog

 IMPORTING

 e\_exit = lv\_answer

 TABLES

 t\_outtab = gt\_labst.

# Web

## Abap 문법

* <http://members.tripod.com/~sap_abap/abapindx.htm>

# 테이블

## 개발

### E070 리퀘스트 상태

### TLOCK: 현재 개발 중 오브젝트.

### DD\* : 딕셔너리 구조

### TRMAC: 글로벌 매크로 설정.

### V\_OP: 배치작업 관련

status = 'R' : 현재 실행중인 배치작업

A - Cancelled/F - Completed/P - Scheduled/R - Active/S - Released

### TPARA: 파라미터 ID

### TSTCP : PARAM 필드에 유지보수 뷰 이름 \* 으로 검색

### RFCDES: RFC 목적지

## 마스터

### T527X: 조직 마스터

## Long Text

### STXH: 테스트 헤더

### STXL: 텍스트 라인(SAP Script 형태)

### STXB: 텍스트 라인(기타 형태)

## 변경문서

### CDHDR: 변경문서 헤더

### CDPOS: 변경문서 품목

# EXIT

### SUSR0001: 로그온 후 처리 됨.

#### EXIT\_SAPLSUSF\_001

ZXUSRU01

### FMAVC\_ENTRY\_FILTER

문서 생성/변경 시 예산 관련

# 모듈 관련

## FI

### 전표유형

### 전기키

차/대변을 결정하고 계정 분류

|  |  |
| --- | --- |
| 구분 | 차/대변 |
| 고객 | 01~10 차변11~20 대변 |
| 업체 | 21~30 차변31~40 대변 |

# 기타

## VB등에서 RFC 함수 호출하기 위해 필요한 OCX

기본적으로 SAPGui를 설치하면 OCX는 등록된다.

단, wdtfuncs.ocx 만 등록이 되어 있지 않아 등록을 해 주어야만 한다.

ActiveX controls들을 등록하기 위해서 다음과 같이 한다.

\*\* Windows 95 ~ 98 환경

1. \\windows\system 디렉터리 아래의 파일을 Copy한다.

"librfc32.dll" "msvcp60.dll" => 꼭 시스템 디렉터리에 있어야 한다.

2. wdtfuncs.ocx를 등록하기 전에 wdtaocx.ocx를 등록한다.

function control을 사용해서 함수를 부르기 위해 필요한 것이 있는데 wdtaocx.ocx이 그 작업을 한다.

3. Function만 사용한다면, wdtlog.ocx(Logon), wdtfuncs.ocx(Function)만 있으면 된다

=> 시스템 디렉터리에 복사한 후 레지스트리에 등록한다.