

# iSQL User's Manual

---

## Altibase 7.1

Altibase® Tools & Utilities



Altibase Tools & Utilities iSQL User's Manual

Copyright © 2001~2023 Altibase Corp. All Rights Reserved.

본 문서의 저작권은 (주)알티베이스에 있습니다. 이 문서에 대하여 당사의 동의없이 무단으로 복제 또는 전용할 수 없습니다.

**(주)알티베이스**

08378 서울시 구로구 디지털로 306 대륭포스트타워II 10층

전화 : 02-2082-1114

팩스 : 02-2082-1099

고객서비스포털 : <http://support.altibase.com>

홈페이지 : <http://www.altibase.com>

# 목차

---

- [서문](#)
  - [이 매뉴얼에 대하여](#)
- [1.iSQL 이용방법](#)
  - [iSQL의 개요](#)
  - [iSQL 설정](#)
  - [iSQL 커맨드 라인 옵션](#)
  - [iSQL 명령어](#)
  - [iSQL 관련 환경변수](#)
  - [개인별 iSQL 환경 설정](#)
- [2.iSQL 사용 예](#)
  - [로그인](#)
  - [Altibase의 구동 및 종료](#)
  - [접속 연결 및 해제](#)
  - [데이터베이스와 객체 정보 조회](#)
  - [트랜잭션 제어](#)
  - [파일 관리](#)
  - [SELECT 결과 포매팅](#)
  - [출력 옵션](#)
  - [iSQL 화면 설정 보기](#)
  - [호스트 변수](#)
  - [PREPARE SQL문 수행](#)
  - [프로시저 생성과 실행 및 삭제](#)
  - [함수 생성과 실행 및 삭제](#)
  - [사용자 편의 기능](#)
  - [내셔널 캐릭터 사용법](#)

# 서문

---

## 이 매뉴얼에 대하여

이 매뉴얼은 데이터베이스에 접속해 데이터베이스 정보와 서버의 정보를 조회하고 제어할 수 있는 도구인 iSQL의 사용법에 대해 설명한다.

## 대상 사용자

이 매뉴얼은 다음과 같은 Altibase 사용자를 대상으로 작성되었다.

- 데이터베이스 관리자
- 성능 관리자
- 데이터베이스 사용자
- 응용 프로그램 개발자
- 기술지원부

다음과 같은 배경 지식을 가지고 이 매뉴얼을 읽는 것이 좋다.

- 컴퓨터, 운영 체제 및 운영 체제 유틸리티 운용에 필요한 기본 지식
- 관계형 데이터베이스 사용 경험 또는 데이터베이스 개념에 대한 이해
- 컴퓨터 프로그래밍 경험
- 데이터베이스 서버 관리, 운영 체제 관리 또는 네트워크 관리 경험

## 소프트웨어 환경

이 매뉴얼은 데이터베이스 서버로 Altibase 버전 7.1을 사용한다는 가정 하에 작성되었다.

## 이 매뉴얼의 구성

이 매뉴얼은 다음과 같이 구성되어 있다.

- 제 1장 iSQL 이용방법  
이 장은 iSQL의 개요와 제공하는 명령어 및 사용방법에 대해 설명한다.
- 제 2장 iSQL 사용 예  
이 장은 iSQL이 제공하는 각각의 명령어 대해 자세한 사용 예를 들어 설명한다.

## 문서화 규칙




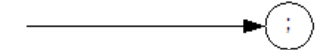
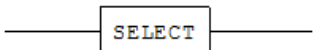
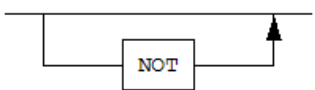
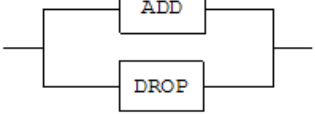
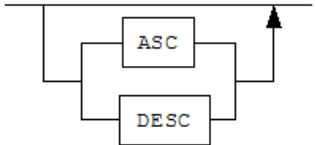
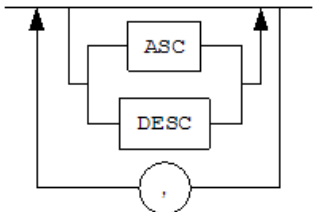
이 절에서는 이 매뉴얼에서 사용하는 규칙에 대해 설명한다. 이 규칙을 이해하면 이 매뉴얼과 설명서 세트의 다른 매뉴얼에서 정보를 쉽게 찾을 수 있다.

여기서 설명하는 규칙은 다음과 같다.

- 구문 다이어그램
- 샘플 코드 규칙

## 구문 다이어그램

이 매뉴얼에서는 다음 구성 요소로 구축된 다이어그램을 사용하여, 명령문의 구문을 설명한다.

구성 요소	의미
	명령문이 시작한다. 완전한 명령문이 아닌 구문 요소는 화살표로 시작한다.
	명령문이 다음 라인에 계속된다. 완전한 명령문이 아닌 구문 요소는 이 기호로 종료한다.
	명령문이 이전 라인으로부터 계속된다. 완전한 명령문이 아닌 구문 요소는 이 기호로 시작한다.
	명령문이 종료한다.
	필수 항목
	선택적 항목
	선택사항이 있는 필수 항목. 한 항목만 제공해야 한다.
	선택사항이 있는 선택적 항목
	선택적 항목. 여러 항목이 허용된다. 각 반복 앞부분에逗마가 와야 한다.

## 샘플 코드 규칙

코드 예제는 SQL, Stored Procedure, iSQL 또는 다른 명령 라인 구문들을 예를 들어 설명한다.

아래 테이블은 코드 예제에서 사용된 인쇄 규칙에 대해 설명한다.

규칙	의미	예제
[ ]	선택 항목을 표시	VARCHAR [(size)] [[FIXED   ] VARIABLE]
{ }	필수 항목 표시. 반드시 하나 이상을 선택해야 되는 표시	{ ENABLE   DISABLE   COMPILE }

규칙	의미	예제
	선택 또는 필수 항목 표시의 인자 구분 표시	{ ENABLE   DISABLE   COMPILE } [ ENABLE   DISABLE   COMPILE ]
...	그 이전 인자의 반복 표시 예제 코드들의 생략되는 것을 표시	SQL> SELECT ename FROM employee; ENAME ----- SWNO HJNO HSCHOI ... 20 rows selected.
그 밖 에 기 호	위에서 보여진 기호 이 외에 기호들	EXEC :p1 := 1; acc NUMBER(11,2);
기 울 임 꼴	구문 요소에서 사용자가 지정해야 하는 변수, 특수한 값을 제공해야만 하는 위치	SELECT * FROM <i>table_name</i> ; CONNECT <i>userID/password</i> ;
소 문 자	사용자가 제공하는 프로그램의 요소들, 예를 들어 테이블 이름, 칼럼 이름, 파일 이름 등	SELECT ename FROM employee;
대 문 자	시스템에서 제공하는 요소들 또는 구문에 나타나는 키워드	DESC SYSTEM.SYS_INDICES;

## 관련 자료

자세한 정보를 위하여 다음 문서 목록을 참조하기 바란다.

- Installation Guide
- Getting Started Guide
- Administrator's Manual
- Replication Manual
- SQL Reference
- Stored Procedures Manual
- Error Message Reference

## Altibase는 여러분의 의견을 환영합니다.

이 매뉴얼에 대한 여러분의 의견을 보내주시기 바랍니다. 사용자의 의견은 다음 버전의 매뉴얼을 작성하는데 많은 도움이 됩니다. 보내실 때에는 아래 내용과 함께 고객센터포털(<http://support.altibase.com/kr/>)로 보내주시기 바랍니다.

- 사용 중인 매뉴얼의 이름과 버전
- 매뉴얼에 대한 의견
- 사용자의 성함, 주소, 전화번호

이 외에도 Altibase 기술지원 설명서의 오류와 누락된 부분 및 기타 기술적인 문제들에 대해서 이 주소로 보내주시면 정성껏 처리하겠습니다. 또한, 기술적인 부분과 관련하여 즉각적인 도움이 필요한 경우에도 고객센터포털을 통해 서비스를 요청하시기 바랍니다.

여러분의 의견에 항상 감사드립니다.

# 1.iSQL 이용방법

---

## iSQL의 개요

iSQL은 Altibase에 접속하여 SQL 문과 부가적인 여러 명령어를 통해 서버에 저장되어 있는 자료를 얻고, 수정하는 작업을 수행할 수 있는 사용자 도구이다.

## iSQL의 주요 기능

- Altibase의 구동 및 종료 기능  
Altibase 서버의 구동 및 종료 등의 데이터 베이스 관리를 하기 위해서 여러 개의 커맨드라인 명령어를 사용하는 대신에, iSQL 하나의 도구만을 사용하면 된다.
- 데이터베이스 접속 및 해제 기능  
Altibase 구동 이후 데이터베이스 접속 시 다양한 사용자명으로의 접속 및 해제가 가능하다.
- 데이터베이스 객체 정보 조회 기능  
iSQL에서는 데이터베이스 객체에 대한 모든 정보를 SQL문으로 조회할 수 있으며, 주요 객체에 대해서는 간편한 조회 명령어를 지원한다.
- SQL문 수행을 통한 데이터베이스 관리 기능  
iSQL을 사용해 모든 SQL문을 수행할 수 있도록 지원하므로 트랜잭션 제어, 데이터베이스 변경 등을 빠르고 간편하게 수행할 수 있다.
- 사용자 편의 기능  
위의 기능을 쉽고 편리하게 수행할 수 있도록 파일 처리 기능, 에디터 기능, iSQL상에서의 셸 명령어 수행 기능, HISTORY 기능 등을 지원한다.

## iSQL 설정

iSQL은 서버에 접속하기 위해서 다음과 같은 정보가 필요하다.

- ALTIBASE\_HOME  
서버 혹은 클라이언트가 설치된 경로
- server\_name  
Altibase 서버가 구동되어 있는 컴퓨터 서버의 이름(또는 IP 주소)
- port\_no  
TCP, IPC 또는 IPCDA로 접속할 때 사용할 포트 번호
- user\_id  
데이터베이스에 등록된 사용자 ID
- password  
사용자 ID와 일치하는 암호
- NLS\_USE  
데이터 검색 시, 사용자에게 보여주는 문자 집합

ALTIBASE\_HOME은 환경 변수로 설정하도록 되어있으며, 나머지는 커맨드 라인 옵션을 통해서 설정할 수 있도록 되어 있다. (자세한 내용은 “iSQL 커맨드 라인 옵션”을 참고한다.)



ALTIBASE\_HOME은 iSQL을 사용하기 위해서 반드시 설정해야 하는 환경 변수이다. 일반적으로 서버가 설치될 때 자동으로 설정되지만, 클라이언트의 경우에는 사용자가 직접 설정해야 한다. 설정되지 않았을 경우에는 올바르게 동작하지 않을 수 있으므로 실행 전에 올바르게 설정되어 있는지 확인할 것을 권한다.

port\_no와 NLS\_USE는 환경 변수 또는 서버 설정 파일(altibase.properties)을 이용해서 설정할 수도 있다. 세 가지 방법으로 모두 설정되어 있을 경우 적용 우선 순위는 다음과 같다.

1. 커맨드 라인 옵션
2. 환경 변수(ALTIBASE\_PORT\_NO, ALTIBASE\_NLS\_USE)
3. 서버 설정 파일(altibase.properties)

그러므로 이미 설정된 값과 다른 옵션으로 연결하고자 할 경우, 커맨드 라인 옵션을 사용하면 서버 설정 파일이나 환경 변수를 변경하지 않아도 된다.

옵션이 설정되어 있지 않을 경우에는 iSQL이 처음 실행될 때 옵션 입력 프롬프트를 띄우고 사용자에게서 해당 값을 입력 받는다. 이 때 바르지 않은 형식이나 유효하지 않은 값을 입력할 경우, iSQL은 올바르게 동작하지 않을 수도 있다.

특히 NLS\_USE 옵션은 사용자가 설정하지 않았더라도 실행 시에 입력 프롬프트가 나타나지 않는다. 만약 사용자가 NLS\_USE 옵션을 설정하지 않았다면 기본값인 US7ASCII를 이용해 접속을 시도한다. 이 때 데이터베이스의 캐릭터 셋이 US7ASCII이 아닐 경우에는 바르게 실행되지 않거나 사용자 데이터가 일부 깨질 수 있으므로 반드시 NLS\_USE를 사용 환경에 맞는 값으로 설정해야 한다.

원활한 iSQL 사용을 위해 다음 환경 변수를 설정할 것을 권장한다.

- ALTIBASE\_HOME : 서버 혹은 클라이언트가 설치된 경로
- ALTIBASE\_PORT\_NO : 서버에 접속할 때 사용할 포트 번호
- ALTIBASE\_NLS\_USE : 데이터 검색 시, 사용자에게 보여주는 문자 집합
- PATH : 실행 파일이 있는 경로인 \$ALTIBASE\_HOME/bin 추가

## iSQL 커맨드 라인 옵션

iSQL을 실행하기 위해서는 반드시 Altibase 서버를 먼저 구동 시켜야 한다. 옵션은 대문자 또는 소문자 모두 사용할 수 있다.

```
isql
[-H]
[-S server_name]
[-PORT port_no]
[-U user_id] [-P password] [/NOLOG]
[-SYSDBA] [-KEEP_SYSDBA]
[-UNIXDOMAIN-FILEPATH filepath]
[-IPC-FILEPATH filepath]
[-SILENT]
[-F infile_name [param1 [param2]...]] [-O outfile_name] [-NLS_USE nls_name]
[-NLS_NCHAR_LITERAL_REPLACE 0|1]
[-prefer_ipv6] [-TIME_ZONE timezone]
[-ssl_ca CA_file_path | -ssl_capath CA_dir_path]
[-ssl_cert certificate_file_path]
[-ssl_key key_file_path]
[-ssl_verify]
```

- -S *server\_name*

Altibase 서버가 구동되어 있는 컴퓨터 서버의 이름(또는 IP 주소)을 명시한다.

만약 ISQL\_CONNECTION 환경 변수가 IPC 또는 UNIX로 설정되어 있을 때, 이 옵션에 원격 서버를 명시해서 접속을 시도하면, iSQL은 ISQL\_CONNECTION의 설정을 무시하고 TCP로 원격 서버에 접속할 것이다. 이 때, ISQL\_CONNECTION 설정이 무시되었다는 경고 메시지가 출력된다.

IPv4 주소 또는 IPv6 주소를 사용할 수 있다. IPv6 주소는 “[”과 ”]”로 에워싸여야 한다.

예를 들어, localhost 를 명시하고자 할 때, 가능한 값은 다음과 같다.

localhost (호스트 이름), 127.0.0.1 (IPv4주소), [::1] (IPv6주소)

IPv6 주소 표기법에 대한 자세한 내용은 Administrator's Manual을 참고하기 바란다.

- -PORT *port\_no*

TCP, IPC 또는 IPCDA로 접속할 때 해당 포트 번호를 명시한다. 단 유닉스 환경에서 IPC로 접속시 이 옵션은 명시하지 않아야 한다. 만약 명시하면 경고 메시지가 출력된 후, 서버에 접속한다.

TCP로 접속하려면 먼저 클라이언트에서 환경 변수 'ISQL\_CONNECTION=TCP'를 설정하고, 옵션에 PORT\_NO를 입력한다. ISQL\_CONNECTION 환경 변수의 값이 IPC가 아니고 -PORT 옵션을 생략한다면, ALTIBASE\_PORT\_NO와 PORT\_NO 프로퍼티를 차례로 참조하고 모두 설정되어 있지 않다면, 포트 번호 입력 프롬프트가 출력된다.

- -U *user\_id*

데이터베이스에 등록된 사용자 ID를 명시한다.

- -P *password*

사용자 ID와 일치하는 암호를 명시한다.

- /NOLOG

데이터베이스에 접속하지 않고 iSQL을 실행한다.

- -SYSDBA

SYS 사용자가 관리자 모드로 iSQL 유틸리티를 사용하기 위해서 -SYSDBA

옵션을 사용할 수 있다. 서버가 구동되어 있지 않다면, iSQL은 idle 인스턴스로 접속할 것이며, 그 상태에서 서버를 구동할 수 있다.

- -KEEP\_SYSDBA

사용자가 -sysdba 옵션으로 관리자 모드로 접속 시, 서버 구동 후에는 서비스 세션으로 재접속 된다. -keep\_sydba 옵션은 서버 구동 후 서비스 세션으로 재접속하지 않고 관리자 모드를 유지하게 한다.

- -UNIXDOMAIN-FILEPATH *filepath*

유닉스 환경에서 서버와 클라이언트가 유닉스 도메인 소켓으로 접속할 때 (ISQL\_CONNECTION=UNIX), ALTIBASE\_HOME이 서로 다르다면 유닉스 도메인의 소켓 경로가 다르게 되어 접속이 불가능하다. 이 때 서버와 클라이언트가 같은 파일 (e.g. ALTIBASE\_HOME/trc/cm-unix)을 사용하도록 하면, 유닉스 도메인 통신이 가능해진다.

- -IPC-FILEPATH *filepath*

유닉스 환경에서 서버와 클라이언트가 IPC로 접속 (ISQL\_CONNECTION=IPC)할 때, ALTIBASE\_HOME이 서로 다르다면 유닉스 도메인의 소켓 경로가 다르게 되어 접속이 불가능하다. 이 때 ALTIBASE\_HOME/trc/cm-ipc 파일을 이용하면, 유닉스 도메인 통신이 가능해져 공유 메모리의 정보를 가져올 수 있다. 그러나 이 옵션은 환경변수 ALTIBASE\_IPC\_FILEPATH를 설정하였다면, 생략해도 된다.

- **-IPCDA-FILEPATH *filepath***  
유닉스 환경에서 서버와 클라이언트가 IPCDA로 접속 (ISQL\_CONNECTION=IPCDA)할 때, ALTIBASE\_HOME이 서로 다르다면 유닉스 도메인의 소켓 경로가 다르게 되어 접속이 불가능하다. 이 때 ALTIBASE\_HOME/trc/cm-ipcda 파일을 이용하면, 유닉스 도메인 통신이 가능해져 공유 메모리의 정보를 가져올 수 있다. 그러나 이 옵션은 환경변수 IPCDA\_FILEPATH를 설정하였다면 생략해도 된다.
- **-F infile\_name [*param1* [*param2*]...]**  
iSQL 실행 후 바로 실행할 스크립트 파일을 명시한다.  
파일 이름에 특수 문자 또는 공백이 포함된 경우 큰따옴표를 사용해야 한다.  
예) -F \"file name\"  
스크립트 파일의 치환 변수에 대입할 값을 파라미터로 지정할 수 있다. 치환 변수에 대한 설명은 'START 명령에 파라미터 전달'을 참조하기 바란다.
- **-O outfile\_name**  
iSQL 실행 후 실행한 명령들에 대한 결과들을 저장할 파일을 명시한다. 이 파일은 현재 디렉터리에 생성된다. 파일이 기존에 존재할 경우, 기존 내용 위에 겹쳐 쓴다.  
파일 이름에 특수 문자 또는 공백이 포함된 경우 큰따옴표를 사용해야 한다.  
예) -O \"file name\"
- **-H**  
iSQL의 실행 방법을 보여준다.
- **-SILENT s**  
ilent 모드를 켜는 옵션이다. silent 모드를 켜면 Copyright 등의 부가적인 설명들을 보여주지 않는다.
- **-NLS\_USE 데이터 검색 시, 사용자에게 보여주는 문자 집합(Character Set)이다.**  
iSQL을 실행하는 터미널의 encoding을 명시하여 준다. 생략 시 환경변수 ALTIBASE\_NLS\_USE, altibase.properties를 차례로 참조하며, 설정되지 않았을 경우에는 기본 문자 집합 (US7ASCII)을 사용한다.
  - US7ASCII
  - KO16KSC5601
  - MS949
  - BIG5
  - GB231280
  - MS936
  - UTF8
  - SHIFTJIS
  - MS932
- **EUCJP-NLS\_NCHAR\_LITERAL\_REPLACE**  
0 : “N” 문자가 있는지 검사하지 않고 쿼리 문 전체를 데이터베이스 문자 셋으로 변환한다.  
1 : “N” 문자가 붙어있는 NCHAR 리터럴은 데이터베이스 문자 셋으로 변환하지 않는다.
- **-prefer\_ipv6**  
-S 옵션으로 호스트 이름을 입력했을 때, 접속할 IP 주소의 버전을 결정하는 옵션이다.  
이 옵션을 명시하면, 호스트 이름을 IPv6 주소로 바꾸어 접속한다.

이 옵션을 명시하지 않으면, iSQL은 IPv4 주소로 접속한다.  
 선호하는 버전의 IP 주소로의 접속이 실패하면, 다른 IP 버전 주소로 접속을 다시 시도한다.  
 예를 들어, -S 옵션에 "localhost"를 입력하고 이 옵션을 명시하면, iSQL은 처음에 IPv6 주소인 [::1]로 접속하고, 이 접속이 실패하면 IPv4 주소인 127.0.0.1로 접속을 다시 시도한다.

- -TIME\_ZONE *timezone*  
 클라이언트의 타임 존을 설정하는 옵션이다. 이 옵션에 DB\_TZ를 지정하면 데이터베이스 서버와 동일한 타임 존이 사용된다. Asia/Seoul과 같은 타임 존의 이름이나 KST 같은 약어를 사용해서 지정할 수 있으며, +09:00 같은 UTC 오프셋 값을 지정할 수도 있다.  
 이 옵션을 생략하면 ALTIBASE\_TIME\_ZONE 환경 변수에 설정된 타임 존이 클라이언트의 타임 존으로 설정되며, 환경 변수도 설정되지 않았다면 데이터베이스 서버와 동일한 타임 존으로 설정된다.
- -ssl\_ca *CA\_file\_path*  
 접속할 알티베이스 서버의 공개키(public key)가 포함된 CA(인증 기관, Certification Authority) 인증서 파일의 위치를 지정한다.
- -ssl\_capath *CA\_dir\_path*  
 접속할 알티베이스 서버의 공개키가 포함된 CA 인증서 파일이 저장되어 있는 디렉토리를 지정한다.
- -ssl\_cert *certificate\_file\_path*  
 클라이언트 인증서 파일의 위치를 지정한다.
- -ssl\_key *key\_file\_path*  
 클라이언트 개인키 파일의 위치를 지정한다.
- -ssl\_verify  
 이 옵션을 지정하면 클라이언트가 서버로부터 전달받은 인증서를 검증한다.
- -ssl\_cipher *cipher\_list*  
 SSL 암호화를 위해 사용할 알고리즘의 이름 후보들을 지정한다. *General Reference*에서 SSL\_CIPHER\_LIST 프로퍼티를 참고한다.

위의 커맨드 라인 중 -S, -U, -P 옵션이 빠져 있는 경우에는 입력 프롬프트가 출력되어 사용자에게 그 옵션 값을 받는다.

## iSQL 명령어

iSQL을 실행 시키면 iSQL 명령어 실행을 위한 프롬프트가 나오고, 이곳에 iSQL 명령어들을 입력하면 그 결과를 보여주는 형태로 프로그램이 동작한다. 아래의 표에 iSQL 각각의 명령에 대해 설명하였다.

분류	종류	명령어	설명
iSQL 구동 및 종료	구동	\$ isql [option]	셸 상에서 이 명령어를 수행하면 iSQL이 구동된다. 사용 가능한 옵션에 대해서는 iSQL 커맨드 라인 옵션 절의 내용을 참조한다.
	프롬프트	iSQL>	iSQL 프롬프트로 명령어 입력 후 ENTER 키를 입력한다.

	종료	EXIT; QUIT;	iSQL을 종료한다.
Altibase 구동 및 종료	Altibase 구동	STARTUP	PRE-PROCESS, PROCESS, CONTROL, META, SERVICE 중 하나의 옵션을 이용하여 Altibase의 다단계 구동을 수행한다.
	Altibase 종료	SHUTDOWN	NORMAL, IMMEDIATE, ABORT 중 하나의 옵션을 사용하여 Altibase를 종료한다.
데이터베이스 접속 및 해제	다른 사용자 로 서버에 접속	CONNECT [logon] [nls] [AS sysdba]; logon:user1/pass1 nls: NLS=character_set	iSQL에서 데이터베이스 접속 후 다른 사용자로 접속하기 위한 명령어로 패스워드 pass1을 가진 user1이라는 사용자로 접속한다. 접속이 성공하면 이전 세션과 관련된 정보는 지워진다. AS 절은 SYS 사용자가 sysdba 관리자 모드로 서버에 접속하는 것을 허용한다. sysdba로 접속하는 것은 한 사용자만 허용된다. nls 옵션은 문자 집합을 설정한다. 문자집합에 대한 자세한 설명은 위의 절 iSQL 커맨드 라인 옵션: -NLS_USE 옵션을 참조하기 바란다.
	접속해제	DISCONNECT;	현재 세션을 종료하고 서버와의 연결을 끊는다.
데이터베이스 객체 정보 조회	성능 뷰 목록 보기	SELECT * FROM V\$TAB;	시스템이 제공하는 모든 성능 뷰 목록을 보여준다. 이 명령어는 iSQL에서만 사용가능 하다.
	테이블 목록 보기	SELECT * FROM TAB;	현재 생성된 테이블의 목록을 보여준다. 이 명령어는 iSQL에서만 사용가능 하다.
	테이블 구조 보기	DESC samp;	samp 테이블의 구조를 보여준다.
	시퀀스 정보 보기	SELECT * FROM SEQ;	SYS 계정으로 서버에 접속한 경우 모든 시퀀스들의 정보를 보여준다. 일반 사용자로 서버에 접속한 경우는 그 사용자가 생성한 시퀀스들에 대한 정보만 보여준다. 이 명령어는 iSQL에서만 사용 가능하다.
파일 관리	파일에 결과 저장	SPOOL filename;	iSQL에서 실행한 명령의 결과를 filename에 기록하기 시작한다.
		SPOOL OFF;	스풀링을 중지한다.
	sql script 의 실행	START file_name;	script 파일을 읽어, 파일 내의 SQL문들을 순차적으로 수행한다.
		@ file_name;	iSQL 프롬프트 상에서 수행 시 start와 동일한 기능을 갖는다.
		@@ file_name;	스크립트 파일 안에서 사용될 때 호출을 한 스크립트 파일이 위치하는 디렉터리에서 파일을 찾아서 수행한다.

	SQL문 파일 저장	SAVE abc.sql;	현재 iSQL 버퍼에 있는 명령어 중 가장 마지막 명령어가 파일로 저장된다.
	SQL문의 load	LOAD abc.sql;	파일에 저장되어 있는 명령어 중 가장 첫 번째 명령어가 명령어 버퍼의 마지막으로 로드된다.
	DML문을 파일로 저장	SET QUERYLOGGING ON; SET QUERYLOGGING OFF;	INSERT, UPDATE, DELETE, MOVE 등의 DML 문 실행 시 이를 \$ALTIBASE_HOME/trc/isql_query.log에 기록한다. 단, DML문 중 SELECT를 실행한 경우에는 로그에 기록되지 않는다.
	질의문 편집	ED[IT]	가장 최근에 실행된 질의문을 편집한다.
		ED[IT] filename[.sql]	기존 파일 또는 새로운 파일을 편집한다.
		2ED[IT] 또는 2 ED[IT]	히스토리 목록에 있는 번호가 2인 질의문을 편집한다.
	출력 옵션 제어	Select 결과 포매팅	
		SET LINESIZE 100;	select 결과 출력 시 디스플레이 되는 한 라인의 사이즈를 설정한다. 10 에서 32767 사이의 값이어야 한다. 기본값: 80
		SET LOBSIZE 10;	CLOB 칼럼을 출력 시 디스플레이 되는 데이터의 길이를 설정한다. 기본값: 80
		SET LOBOFFSET 3;	CLOB 칼럼을 출력 시 디스플레이 되는 데이터의 오프셋을 설정한다. 기본값: 0
		SET FEED[BACK] ON; SET FEED[BACK] OFF; SET FEED[BACK] n;	쿼리 실행결과 건수의 출력여부를 설정한다.
		SET PAGESIZE 10;	select 결과 레코드들을 몇 개 단위로 출력할지를 결정하는 명령어로 `0`으로 설정할 경우 결과 레코드 전체를 한꺼번에 출력한다. 기본값: 0
		SET HEADING ON; SET HEADING OFF;	select 결과 출력 시 헤더 출력 유무 기본값: ON
		SET COLSIZE N;	CHAR, VARCHAR 타입 칼럼의 결과를 표시할 자릿수 설정. COLSIZE가 LINESIZE에 우선하여 적용된다.

		SET NUM[WIDTH] N;	NUMERIC, DECIMAL, NUMBER, FLOAT 타입의 SELECT 결과를 표시할 자릿수 설정. 기본값: 11
		CL[EAR] COL[UMNS]	COLUMN으로 설정된 칼럼의 형식 해제
		COL[UMN] [{column   expr} [option]]	SELECT 대상(target)이 되는 칼럼의 표시 형식 설정 및 확인
		SET NUMF[ORMAT] format;	NUMERIC, DECIMAL, NUMBER, FLOAT 타입의 SELECT 결과를 표시할 형식 설정
	SQL문 실행시간	SET TIMING ON; SET TIMING OFF;	SQL 명령 실행에 걸린 시간 출력유무 기본값: OFF
	SQL문 실행시간 출력 단위 설정	SET TIMESCALE SEC; SET TIMESCALE MILSEC; SET TIMESCALE MICSEC; SET TIMESCALE NANSEC;	SQL문의 쿼리 수행 시간 단위를 초, 밀리초, 마이크로초, 나노초 등으로 설정한다.
	Check 제약조건 정보 출력의 유무	SET CHKCONSTRAINTS ON; SET CHKCONSTRAINTS OFF;	테이블 구조(DESC)를 볼 때 Check 제약조건 정보 포함 출력 여부 설정. 기본값: OFF
	foreign key 정보 출력의 유무	SET FOREIGNKEYS ON; SET FOREIGNKEYS OFF;	테이블 구조(DESC)를 볼 때 외래 키 정보 포함 출력 여부 설정. 기본값: OFF
	파티션 정보 출력의 유무	SET PARTITIONS ON; SET PARTITIONS OFF;	테이블 구조(DESC)를 볼 때 파티션 정보 포함 출력 여부 설정. 기본값: OFF
	스크립트 실행 결과의 출력 유무	SET TERM ON; SET TERM OFF;	스크립트 파일 실행의 결과 및 명령어를 화면 상에 보여줄지를 결정한다. 기본값: ON
	스크립트 명령어 출력 유무	SET ECHO ON; SET ECHO OFF;	@으로 실행된 스크립트 파일 내의 명령어들의 출력 여부를 설정한다. 기본값 : ON

	치환 변수 대체 여부	SET DEFINE ON; SET DEFINE OFF;	치환 변수가 있는 스크립트 파일 수행 시, 사용자가 입력한 파라미터 값으로 치환 변수를 대체할지 여부를 지정한다. 기본값: OFF
	치환 변수 교체 전후 내용 출력	SET VERIFY ON; SET VERIFY OFF;	치환 변수가 있는 스크립트 파일 수행 시, 치환 변수가 파라미터 값으로 교체되기 전후의 SQL문을 출력할지 여부를 지정한다. 기본값: ON
	실행 계획 트리 출력	ALTER SESSION SET EXPLAIN PLAN = ON; ALTER SESSION SET EXPLAIN PLAN = ONLY; ALTER SESSION SET EXPLAIN PLAN = OFF;	SELECT문에 대한 실행 계획의 출력 여부를 설정한다. 기본값: OFF
	SELECT 결과 출력 방향	SET VERTICAL ON; SET VERTICAL OFF;	레코드를 조회할 때 이 값을 ON으로 설정하면, SELECT의 결과가 세로로 보여진다. 기본값: OFF
	iSQL 화면 설정 값 보기	SHOW LINESIZE	현재의 LINESIZE 값을 보여준다.
		SHOW COLSIZE	현재의 COLSIZE 값을 보여준다.
		SHOW LOBOFFSET	현재의 LOBOFFSET 값을 보여준다.
		SHOW LOBSIZE	현재의 LOBSIZE 값을 보여준다.
		SHOW PAGESIZE	현재의 PAGESIZE 값을 보여준다.
		SHOW PLANCOMMIT	AUTOCOMMIT OFF 모드에서 명령어를 수행할 때 자동으로 커밋되는 여부를 보여준다.
		SHOW QUERYLOGGING	DML 문이 실행될 때 \$ALTIBASE_HOME/trc/isql_query.log에 기록되는지 여부를 보여준다.
		SHOW FEEDBACK	현재 설정된 FEEDBACK 값을 보여준다.
		SHOW HEADING	현재의 HEADING 설정 여부를 보여준다.
		SHOW TERM	현재 TERM 설정 여부를 보여준다.
		SHOW ECHO	현재 ECHO 설정 여부를 보여준다.
		SHOW TIMING	현재의 TIMING 설정 여부를 보여준다.
		SHOW TIMESCLAE	현재의 SQL문의 쿼리 수행 시간 단위가 무엇으로 설정되었는지를 보여준다.
		SHOW USER	현재 사용자를 보여준다.



		SHOW CHKCONSTRAINTS	현재의 Check 제약조건 설정 여부를 보여준다.
		SHOW FOREIGNKEYS	현재의 외래 키 설정 여부를 보여준다.
		SHOW PARTITIONS	현재의 파티션 출력 여부 설정을 보여준다.
		SHOW VERTICAL	현재의 SELECT 결과가 세로로 출력되는지 여부를 보여준다.
		SHOW ALL	현재 세션의 화면 설정 값을 보여준다.
변수 및 Prepared SQL문	변수 선언	VAR p1 INTEGER;	INTEGER 타입의 변수 p1을 선언한다.
		VARIABLE p2 CHAR(10);	CHAR 타입의 변수 p2를 선언한다.
	변수에 값 할당	EXECUTE :p1 := 100;	변수 p1에 100을 할당한다.
		EXEC :p2 := 'abc';	변수 p2에 'abc'를 할당한다.
	변수 보기	PRINT VAR[IABLE];	현재 선언된 변수들을 보여준다.
		PRINT p1;	변수 p1의 타입과 값을 보여준다.
	Prepared SQL문 수 행	PREPARE SQL문;	Prepared SQL문으로 질의 최적화 과정과 실행 과정을 나누어 수행하게 한다. iSQL에서의 SQL문 수행은 기본적으로 최적화와 실행을 한번에 수행하는 Direct Execution 방법이다. iSQL 상에서 두 가지 수행 방법에 대한 결과에는 차이가 없으며 Prepared SQL문의 경우 변수를 사용해 값을 바인딩 하여 SQL문 수행이 가능하다.
사용자 편의 기능	히스토리 목록 보기	HISTORY; H;	현재 iSQL buffer에 저장되어 있는 명령어들의 목록을 보여준다.
	반복 실행	/	현재 iSQL buffer의 명령어를 반복하여 실행한다. 가장 최근에 수행한 명령어가 다시 실행된다.
		2/	HISTORY 명령에 의해 나타난 목록의 번호가 2인 명령어가 실행된다.
	셸 명령 실행	! shell command	느낌표 다음에 셸 명령을 입력하면 iSQL에서 바로 셸 명령이 실행된다.
	명령 프롬 프트 변경	SET SQLP[ROMPT] {text}	iSQL 명령 프롬프트를 설정한다.
	주석	/* comment */ -- comment	여러 라인 주석 한 라인 주석

	도움말	HELP; HELP INDEX; HELP EXIT;	도움말 사용법 명령어 리스트 출력 EXIT 명령어에 대한 설명
--	-----	------------------------------------	--

## iSQL 관련 환경변수

### ALTIBASE\_HOME

패키지가 설치된 디렉터리의 위치이다.

ALTIBASE\_HOME은 iSQL을 사용하기 위해서 반드시 설정해야 하는 환경 변수이다. 일반적으로 서버가 설치될 때 자동으로 설정되지만 클라이언트의 경우에는 서버의 환경 변수와 충돌이 있을 수 있으므로 사용자가 직접 설정해야 한다.

### ALTIBASE\_PORT\_NO

접속할 서버의 포트 번호이다. -PORT 옵션 또는 altibase.properties 파일 내의 프로퍼티를 통해서 지정할 수도 있다.

포트 번호 설정의 우선 순위는 -PORT 옵션, 환경변수 ALTIBASE\_PORT\_NO, altibase.properties 파일 내의 프로퍼티 순이며 아무것도 설정되지 않았을 경우에는 포트 번호 입력 프롬프트가 출력된다.

### ALTIBASE\_SSL\_PORT\_NO

iSQL이 SSL/TLS 통신으로 접속할 서버의 포트 번호이다.

SSL 포트 번호의 우선 순위는 -PORT 옵션, 환경변수, ALTIBASE\_SSL\_PORT\_NO, altibase.properties 파일 내의 프로퍼티 순이다. 만약 아무것도 설정되지 않았을 경우에는 포트 번호 입력 프롬프트가 출력된다.

### ALTIBASE\_NLS\_USE

데이터 검색 시 사용자에게 보여주기 위해 사용되는 문자 집합이다.

- US7ASCII
- KO16KSC5601
- MS949
- BIG5
- GB231280
- MS936
- UTF8
- SHIFTJIS
- MS932
- EUCJP

-NLS\_USE 옵션 또는 altibase.properties파일 내의 프로퍼티를 통해서 지정할 수도 있다.

NLS\_USE 설정의 우선 순위는 -NLS\_USE 옵션, 환경 변수 ALTIBASE\_NLS\_USE, altibase.properties 파일 내의 프로퍼티 순이며 설정되지 않았을 경우에는 기본 문자 집합(US7ASCII)을 사용한다.

## ALTIBASE\_NLS\_NCHAR\_LITERAL\_REPLACE

기본적으로 클라이언트는 쿼리 문 전체를 데이터베이스 문자 셋으로 변환하여 전송한다. 그러나, 특정 리터럴에 대해 이런 동작을 막으려면, 이 환경 변수의 값을 1로 설정하고 그 리터럴 앞에 "N" 문자를 덧붙이면 된다. 즉, NCHAR 리터럴로 만드는 것이다.

이 환경 변수의 값이 1일 때, 클라이언트는 쿼리 문 내의 모든 리터럴 앞에 "N" 문자가 있는지 찾는다. 만약 찾게 되면 클라이언트는 그 리터럴을 데이터베이스 문자 셋으로 변환하지 않고 그대로 전송하며 서버가 직접 내셔널 문자 셋으로 변환한다. 이것은 데이터베이스 문자 셋과는 다른 인코딩이 필요한 NCHAR 타입 데이터를 사용하고자 할 때 유용하다.

- 0: "N" 문자가 있는지 검사하지 않고 쿼리 문 전체를 데이터베이스 문자 셋으로 변환한다.
- 1: "N" 문자가 붙어있는 NCHAR 리터럴은 데이터베이스 문자 셋으로 변환하지 않는다.

이 값을 1로 설정하는 것은 클라이언트의 비용이 크게 발생하므로, 사용시 주의가 필요하다.

## ISQL\_CONNECTION

Altibase를 클라이언트-서버 구조로 운영할 때, 사용자는 응용 시스템의 구성에 적합한 클라이언트-서버 프로토콜을 선택하여 환경 변수를 설정할 수 있다.

Altibase는 TCP/IP, IPC, IPCDA와 UNIX DOMAIN 소켓, SSL/TLS 프로토콜, 인피니밴드(Infiniband)를 제공한다.

Altibase 서버와 통신하기 위한 기본값은 TCP/IP 프로토콜이다.

- TCP
- UNIX
- IPC
- IPCDA
- SSL
- IB

단, IPC 또는 IPCDA 프로토콜을 이용할 경우엔 Altibase 프로퍼티들 중에서 IPC 채널과 관련된 프로퍼티들의 값(IPC\_CHANNEL\_COUNT 또는 IPCDA\_CHANNEL\_COUNT)도 함께 고려해야 한다.

예) IPC 사용시 환경 변수 설정

```
CSH: setenv ISQL_CONNECTION IPC
SH: ISQL_CONNECTION=IPC; export ISQL_CONNECTION
```

주의: ISQL\_CONNECTION 환경 변수의 설정 값이 UNIX 또는 IPC인 경우, -s 옵션에 원격 서버를 명시해서 iSQL을 실행하면 ISQL\_CONNECTION 설정이 무시되었다는 경고 메시지와 함께 TCP로 원격 서버에 접속한다.

## ISQL\_BUFFER\_SIZE

쿼리를 저장할 버퍼의 크기를 환경변수를 이용하여 지정할 수 있다.

```
CSH: setenv ISQL_BUFFER_SIZE 128000
SH: ISQL_BUFFER_SIZE = 128000; export ISQL_BUFFER_SIZE
```

## ALTIBASE\_DATE\_FORMAT

Date 데이터 타입인 data를 select 시 기본 날짜 형식인 YYYY/MM/DD HH:MI:SS을 환경변수 ALTIBASE\_DATE\_FORMAT을 설정하여 새로운 날짜 형식으로 나타낼 수 있다.

예) Born, Korn, 또는 Bash Shell의 경우

```
export ALTIBASE_DATE_FORMAT='DD-MON-YYYY'
```

## ISQL\_EDITOR

기본 편집기(예: /bin/vi)를 바꾸기 위한 환경변수를 설정할 수 있다.

```
CSH: setenv ISQL_EDITOR /usr/bin/ed
SH: ISQL_EDITOR=/usr/bin/ed; export ISQL_EDITOR
```

## ALTIBASE\_IPC\_FILEPATH

유닉스 환경에서 서버와 클라이언트가 IPC로 접속할 때 ALTIBASE\_HOME이 다른 경우, 유닉스 도메인의 소켓 경로가 일치하지 않아 접속할 수 없다. 이 때 클라이언트 측의 ALTIBASE\_IPC\_FILEPATH 환경 변수 또는 iSQL 옵션인 -IPC-FILEPATH를 서버의 \$ALTIBASE\_HOME/trc/cm-ipc파일로 설정하여 서버와 클라이언트가 같은 소켓 파일을 사용하도록 하면, IPC 접속이 가능하다.

## IPCD\_A\_FILEPATH

유닉스 환경에서 서버와 클라이언트가 IPCDA로 접속할 때 ALTIBASE\_HOME이 다른 경우, 유닉스 도메인의 소켓 경로가 일치하지 않아 접속할 수 없다. 이 때 클라이언트 측의 IPCDA\_FILEPATH 환경 변수 또는 iSQL 옵션인 -IPCD\_A-FILEPATH를 서버의 \$ALTIBASE\_HOME/trc/cm-ipcd파일로 설정하여 서버와 클라이언트가 같은 소켓 파일을 사용하도록 하면, IPCDA 접속이 가능하다.

## ALTIBASE\_TIME\_ZONE

클라이언트의 타임 존을 설정하는 환경 변수이다. 이 환경 변수에 DB\_TZ를 지정하면 데이터베이스 서버와 동일한 타임 존이 사용된다.

이 환경 변수는 Asia/Seoul과 같은 타임 존의 이름이나 KST 같은 약어를 사용해서 설정할 수 있다. 또는 +09:00 같은 UTC 오프셋 값을 사용해서 설정할 수도 있다.

## ALTIBASE\_UT\_FILE\_PERMISSION

aexport, iLoader, iSQL이 생성하는 파일들의 권한을 설정하는 공통 환경변수이다.

값을 설정하지 않으면 666 ( user:rw, group:rw, other: rw)로 설정된다.

예) user:rw, group:--, other:--로 설정하는 경우,  
export ALTIBASE\_UT\_FILE\_PERMISSION=600

ISQL\_FILE\_PERMISSION, AEXPORT\_FILE\_PERMISSION, 또는 ILO\_FILE\_PERMISSION이 설정된 경우, ALTIBASE\_UT\_FILE\_PERMISSION 환경 변수 보다 우선 처리된다.

예)export ALTIBASE\_UT\_FILE\_PERMISSION=660; export ISQL\_FILE\_PERMISSION=600;  
iSQL에서 생성되는 파일의 권한은 ISQL\_FILE\_PERMISSION=600이 우선처리되어 user:rw, group:--, other:--으로 설정된다.  
aexport, iloader가 생성하는 파일의 권한은 ALTIBASE\_UT\_FILE\_PERMISSION=660에 따라 user:rw, group:rw, other:--으로 설정된다.

## ISQL\_FILE\_PERMISSION

iSQL이 생성하는 파일 권한을 설정하는 환경 변수이다. 값을 설정하지 않으면 666 ( user:rw, group:rw, other: rw)로 설정된다.

예) user:rw, group:--, other:--로 설정하는 경우,  
export ISQL\_FILE\_PERMISSION=600

## 개인별 iSQL 환경 설정

iSQL 사용자들은 특별한 방법으로 iSQL 환경을 설정하고 각 세션에서 그러한 설정을 다시 사용할 수 있다. 예를 들어, OS 파일을 통하여 질의 결과마다 현재 시간을 출력할 수 있도록 사용자가 원하는 출력 서식 형태로 만들 수 있다. 이러한 파일은 다음과 같이 두 가지로 나눌 수 있다.

### glogin.sql

iSQL 시작 시의 초기화 작업을 위하여 DB 관리자에 의해 생성된 전역 스크립트 파일인 glogin.sql의 사용을 지원한다. iSQL은 임의의 사용자가 iSQL을 기동할 때 마다 이 스크립트를 실행한다. 전역 파일은 DB 관리자가 모든 사용자들에게 특별한 사이트에서 iSQL 환경을 설정할 수 있도록 한다. 전역 스크립트 파일은 \$ALTIBASE\_HOME/conf 밑에 위치한다.

### login.sql

iSQL은 또한 glogin.sql 후에 실행되는 login.sql 파일도 지원한다. 만약, glogin.sql 파일과 login.sql 파일이 모두 존재하는 경우 iSQL 구동 시 glogin.sql이 실행된 후 login.sql이 실행되므로 login.sql에 있는 명령어들로 우선수행 (override) 된다.

하나의 유닉스 계정을 여러 명이 사용할 경우에는 glogin.sql 파일을 개인적인 용도로 수정하기가 불가능할 수도 있다. 이런 경우 일반 사용자는 SQL 명령어들, 저장 프로시저, 또는 iSQL 명령어들을 각자의 개인용 작업 디렉터리 내에 login.sql 파일에 첨가할 수 있다. 사용자가 iSQL을 구동할 때, iSQL은 자동적으로 현재 디렉토리에서 login.sql 파일을 찾고, 그 안에 명령어들을 수행한다.

login.sql 파일은 iSQL 초기 설정이나 각각의 세션에 대한 동작을 조정할 수 없다.

## LOGIN 파일 변경

사용자는 임의의 다른 스크립트들처럼 LOGIN 파일을 변경할 수 있다. 다음은 임의의 사용자(user1)가 autocommit mode를 off로 변경하고 SQL 문들을 실행하기 위하여 LOGIN 파일을 작성한 예이다.

```
$ vi glogin.sql
AUTOCOMMIT ON
SET HEADING OFF
SELECT sysdate FROM dual;

$ vi login.sql
AUTOCOMMIT OFF;
SET HEADING ON
DROP TABLE savept;
CREATE TABLE savept(num INTEGER);
INSERT INTO savept VALUES(1);
SAVEPOINT sp1;
INSERT INTO savept VALUES(2);
SELECT * FROM savept;
ROLLBACK TO SAVEPOINT sp1;
SELECT * FROM savept;
COMMIT;

$ isql
-----
      Altibase Client Query utility.
      Release Version 7.1.0.1
      Copyright 2000, Altibase Corporation or its subsidiaries.
      All Rights Reserved.
-----
Write Server Name (default:127.0.0.1) :
Write UserID : user1
Write Password :
ISQL_CONNECTION = TCP, SERVER = 127.0.0.1, PORT_NO = 20300
Set autocommit on success.  -> 먼저 glogin.sql을 실행

28-DEC-2004      -> heading off
1 row selected.
Set autocommit off success. -> glogin.sql이 실행된 후 사용자의 현재 작업 디렉터리에서
login.sql을 실행
Drop success.
Create success.
1 row inserted.
Savepoint success.  -> autocommit mode off 에서만 실행 가능
1 row inserted.
SAVEPT.NUM              -> heading on
-----
1
2
2 rows selected.
Rollback success.
SAVEPT.NUM
-----
1
```

```
1 row selected.  
Commit success.
```

## 주의 사항

LOGIN 파일에는 보안상의 이유로 사용자 이름과 비밀번호를 함께 입력하는 CONNECT 명령어를 사용할 수 없다. 만약 LOGIN 파일에 CONNECT 명령어가 포함된 경우 아래의 경고 메시지를 출력하고 해당 명령어는 실행되지 않는다.

```
WARNING: CONNECT command in glogin.sql file ignored
```

## 2.iSQL 사용 예

이 장은 iSQL을 이용하여 데이터베이스를 다루는 몇 가지 예를 설명한다.

### 로그인

iSQL 유틸리티를 사용하기 위해서는 먼저 로그인 과정을 거쳐야 하는데, 커맨드 라인 상에서 직접 연결 정보를 입력하는 방법과 입력 프롬프트 상에서 입력하는 방법이 있다.

```
isql -U userID -P password [-SYSDBA]
또는
isql [-SYSDBA]
```

서버와 연결하기 위한 부가 정보로는 서버 이름(-S), 사용자 ID(-U), 패스워드(-P)가 있다. 사용자 ID와 패스워드는 대소문자를 구별하지 않는다.

SYS 사용자가 관리자 모드로 iSQL 유틸리티를 사용하기 위해서는 -SYSDBA 옵션을 사용한다. SYSDBA 옵션으로 원격에서도 접속이 가능하다.

사용자 ID에 특수 문자 또는 공백이 포함된 경우 큰따옴표를 사용해야 한다.

```
$ isql -U \"user name\"
```

### 제한 사항

- SYSDBA 모드로 접속하는 것은 한 명의 사용자만 허용된다. 2명 이상의 사용자가 동시에 SYSDBA 모드로 접속할 수 없다.
- 원격에서 SYSDBA 모드로 접속할 수 있지만, DBMS를 구동할 수는 없다.

시스템 권한에 대한 자세한 정보는 *SQL Reference* 을 참조하기 바란다.

iSQL 사용 중 발생하는 에러에 대한 자세한 정보는 Error Message Reference를 참조하기 바란다.

```
$ isql -U sys -P manager [-SYSDBA]
```



```
$ isql [-sysdba]
```

```
-----
Altibase Client Query utility.
Release Version 7.1.0.1
Copyright 2000, Altibase Corporation or its subsidiaries.
All Rights Reserved.
-----
```

```
Write Server Name (default:127.0.0.1) :
```

```
Write UserID : sys
```

```
Write Password : manager -> 화면에는 암호가 나타나지 않는다.
```

```
ISQL_CONNECTION = TCP, SERVER = 127.0.0.1, PORT_NO = 20300
```

```
isql(sysdba)> -> isql이 서버와 연결된 상태이며, 여기에서 SQL, isql, PSM 명령등
을 입력하여 실행할 수 있다.
```

## Altibase의 구동 및 종료

Altibase의 구동 및 종료는 iSQL을 사용해 수행한다.

### Altibase 구동

Altibase를 구동시키기 위해서는 데이터베이스 생성 시와 마찬가지로 우선 iSQL을 `-sysdba` 옵션으로 띄워야 한다.

Altibase의 startup 명령어는 Altibase(iSQL 포함)를 설치한 유닉스 계정으로만 수행이 가능하다.

다음은 iSQL를 이용한 Altibase 구동 예제이며 Altibase 구동에 대한 자세한 설명은 *Administrator's Manual* 제2장 Altibase 구동 및 종료의 내용을 참조한다.

```
$ isql -s 127.0.0.1 -u sys -p manager -sysdba
```

```
-----
Altibase Client Query utility.
Release Version 7.1.0.1
Copyright 2000, Altibase Corporation or its subsidiaries.
All Rights Reserved.
-----
```

```
ISQL_CONNECTION = TCP, SERVER = 127.0.0.1, PORT_NO = 20300
```

```
[Connected to idle instance]
```

```
isql(sysdba)> startup service
```

```
Connecting to the DB server... Connected.
```

```
TRANSITION TO PHASE : PROCESS
```

```
TRANSITION TO PHASE : CONTROL
```

```
TRANSITION TO PHASE : META
```

```
[SM] Recovery Phase - 1 : Preparing Database
```

```
: Dynamic Memory Version => Parallel Loading
```

```
[SM] Recovery Phase - 2 : Loading Database
```

```
[SM] Recovery Phase - 3 : Skipping Recovery & Starting Threads...
```

```
Refining Disk Table
```

```
[SM] Refine Memory Table :
..... [SUCCESS]
[SM] Rebuilding Indices [Total Count:100] .....
[SUCCESS]
TRANSITION TO PHASE : SERVICE
[CM] Listener started : TCP on port 20300
[CM] Listener started : UNIX
[RP] Initialization : [PASS]
--- STARTUP Process SUCCESS ---
Command execute success.
```

## Altibase 종료

현재 구동중인 Altibase 서버를 종료하려면 SHUTDOWN 명령어를 사용한다.

다음은 iSQL를 이용한 Altibase 종료 예제이며 Altibase 종료에 대한 자세한 설명은 *Administrator's Manual* 제2장 Altibase 구동 및 종료의 내용을 참조한다.

```
iSQL(sysdba)> shutdown normal
Ok..Shutdown Proceeding....

TRANSITION TO PHASE : Shutdown Altibase
[RP] Finalization : PASS
shutdown normal success.
```

## 접속 연결 및 해제

### 접속 연결(CONNECT)

명시된 사용자 ID로 Altibase에 연결한다. 첫 연결 실패 시 CONNECT 명령어는 사용자 ID나 패스워드를 다시 프롬프트(prompt) 하지 않는다.

```
CONNECT [logon][nls] [AS SYSDBA];
logon: userID[/password]
nls: NLS=character_set
```

- userID/password  
Altibase에 연결하고자 하는 사용자의 id와 패스워드
- NLS=character\_set  
문자집합

```
iSQL> CONNECT sys/manager NLS=US7ASCII
Connect success.
```

- AS SYSDBA

AS 절은 SYS 사용자가 sysdba 관리자 모드로 서버에 접속하는 것을 허용한다.

CONNECT가 성공하면 현재의 세션을 종료하고 명시된 사용자 ID와 패스워드, altibase.properties 내의 프로퍼티 정보를 사용해서 서버에 접속한다. 따라서 이전의 세션 정보는 없어진다.

예를 들어 altibase.properties의 AUTOCOMMIT 모드가 TRUE였고, iSQL에서 AUTOCOMMIT 모드를 FALSE로 변경하여 쿼리를 수행하다가 이 CONNECT문을 수행하였다면 AUTOCOMMIT 모드는 altibase.properties 의 AUTOCOMMIT 프로퍼티에 의하여 TRUE로 변경된다.

CONNECT가 실패한다면 이전의 세션은 종료되고 서버와의 연결이 끊어진 상태가 된다.

즉, 이후에 수행되는 SQL문의 결과는 모두 "Not connected." 이다. 만약, 다시 서버와 연결을 시도할 때는 CONNECT *userID/password* [AS SYSDBA];를 수행한다.

```
$ isql
-----
      Altibase Client Query utility.
      Release Version 7.1.0.1
      Copyright 2000, Altibase Corporation or its subsidiaries.
      All Rights Reserved.
-----

Write Server Name (default:127.0.0.1) :
Write UserID : SYS
Write Password :
ISQL_CONNECTION = TCP, SERVER = 127.0.0.1, PORT_NO = 20300
iSQL> SHOW USER;
User : SYS
iSQL> CREATE USER altiadmin IDENTIFIED BY altiadmin1234;
Create success.
iSQL> CONNECT altiadmin/altiadmin1234;
Connect success.
iSQL> SHOW USER;
User : ALTIADMIN
iSQL> CREATE TABLE altitbl(i1 INTEGER, i2 CHAR(5));
Create success.
iSQL> SELECT * FROM tab;
TABLE NAME                                TYPE
-----
ALTITBL                                  TABLE
.
.
.
33 row selected.
iSQL> CONNECT sys/manager;
Connect success.
iSQL> SHOW USER;
User : SYS
iSQL> CREATE TABLE systbl(i1 INTEGER, i2 CHAR(5));
Create success.
iSQL> SELECT * FROM tab;
USER NAME    TABLE NAME    TYPE
-----
SYSTEM_     SYS_COLUMNS_   SYSTEM TABLE
SYSTEM_     SYS_CONSTRAINTS_   SYSTEM TABLE
.
.
```

```
.  
ALTIADMIN    ALTITBL      TABLE.  
SYS          SYSTBL      TABLE  
.  
.  
.  
93 rows selected.
```

### 주의사항

사용자 이름에 소문자, 특수 문자 또는 공백이 포함된 경우 큰따옴표를 사용해야 한다.

```
iSQL\> CONNECT "user name";
```

## SSL 접속

### 서버 전용 모드

서버 전용 모드(SSL\_CLIENT\_AUTHENTICATION 프로퍼티 값이 0으로 설정)에서 개인 인증서를 사용하는 경우, 서버에서 클라이언트의 인증을 수행하지 않기 때문에 클라이언트 인증서와 개인키 파일의 위치를 지정할 필요가 없다.

서버에서 제공하는 인증서를 검증하기 위해서 -ssl\_verify 옵션을 활성화시키고, 서버의 공개키(public key)가 포함된 CA 인증서 파일의 위치를 지정해야 한다.

```
$ export ISQL_CONNECTION=SSL  
$ isql -s localhost -u sys -p MANAGER  
or  
$ isql -s localhost -u sys -p MANAGER -ssl_verify -ssl_ca ~/cert/ca-cert.pem
```

### 상호 인증 모드

상호 인증 모드(SSL\_CLIENT\_AUTHENTICATION 프로퍼티 값이 1로 설정)로 개인 인증서를 사용하는 경우, 서버에서 클라이언트 인증을 수행한다. 따라서 서버에 제공하는 클라이언트의 인증서와 개인키 파일의 위치를 지정해야 한다.

서버에서 제공하는 인증서를 검증하기 위해서는 -ssl\_verify 옵션을 활성화시키고, 서버의 공개키가 포함된 CA 인증서 파일의 위치를 지정해야 한다.

```
$ export ISQL_CONNECTION=SSL  
$ isql -s localhost -u sys -p MANAGER \  
-ssl_cert ~/cert/client-cert.pem \  
-ssl_key ~/cert/client-key.pem  
or  
$ isql -s localhost -u sys -p MANAGER \  
-ssl_verify -ssl_ca ~/cert/ca-cert.pem \  
-ssl_cert ~/cert/client-cert.pem \  
-ssl_key ~/cert/client-key.pem
```

## 접속 해제

현재 세션을 종료하고 서버와의 연결을 끊는다. 이후에 수행되는 SQL문의 결과는 모두 "Not connected."이며, 다시 서버와 연결을 시도할 때는 CONNECT userID/password;를 수행한다.

```
DISCONNECT;
iSQL> INSERT INTO systbl VALUES(1, 'A1');
1 row inserted.
iSQL> INSERT INTO systbl VALUES(2, 'A2');
1 row inserted.
iSQL> SELECT * FROM systbl;
SYSTBL.I1    SYSTBL.I2
-----
1            A1
2            A2
2 rows selected.
iSQL> DISCONNECT;
Disconnect success.
iSQL> INSERT INTO systbl VALUES(3, 'A3');
[ERR-91020 : No Connection State]
iSQL> SELECT * FROM systbl;
[ERR-91020 : No Connection State]
iSQL> CONNECT sys/manager;
Connect success.
```

## NOLOG 옵션으로 iSQL 실행

'NOLOG' 옵션은 대상 데이터베이스에 접속하지 않고 iSQL을 실행하기 위해 제공된다. 이 옵션을 사용하려면, 서버에 접속하지 않더라도 서버 IP 주소와 포트 번호를 반드시 입력해야 한다.

```
iSQL -s localhost -port 20300 /NOLOG
```

iSQL을 구동한 후 SQL 구문을 수행하려면, CONNECT 명령어로 대상 데이터베이스 사용자 계정과 비밀번호를 입력하여 데이터베이스에 접속한 이후 가능하다.

## 데이터베이스와 객체 정보 조회

### 성능 뷰 조회

성능 뷰는 서버의 상태 및 데이터베이스 정보를 조회할 수 있는 데이터 디렉터리 테이블의 일종으로 Altibase가 제공하는 성능 뷰의 목록을 확인하기 위해서는 다음 명령어를 사용한다.

```
isQL> SELECT * FROM V$TAB;
TABLE NAME                                TYPE
-----
V$ALLCOLUMN                               PERFORMANCE VIEW
V$ARCHIVE                                 PERFORMANCE VIEW
V$BUFFPOOL_STAT                           PERFORMANCE VIEW
V$DATABASE                                PERFORMANCE VIEW
V$DATAFILES                               PERFORMANCE VIEW
V$DISKGC                                   PERFORMANCE VIEW
V$DISKTBL_INFO                             PERFORMANCE VIEW
V$FLUSHINFO                               PERFORMANCE VIEW
```

Altibase가 제공하는 전체 성능 뷰의 목록과 칼럼의 의미 등은 *General Reference*의 데이터 디렉터리 설명을 참조한다.

각 성능 뷰의 데이터는 일반 테이블 조회와 동일한 SELECT문을 사용하여 조회할 수 있으며 조인 등을 사용해 다양한 형태로 결과를 출력할 수 있다.

## 테이블 목록 보기

데이터베이스에 존재하는 모든 테이블에 대한 정보를 알고 싶으면 아래와 같은 명령을 사용하면 된다. sys\_tables\_ 메타 테이블은 Altibase에서 제공하는 데이터베이스 카탈로그 정보를 수록하는 시스템 내부 테이블이다.

```
isQL> SELECT * FROM system_.sys_tables_;
.
.
isQL> SELECT * FROM tab; -> 이 명령어는 isQL에서만 사용가능.
USER NAME    TABLE NAME  TYPE
-----
.
..
```

## 테이블 구조 보기

사용자가 생성한 테이블에 관한 정보를 알고 싶으면 아래와 같은 명령을 사용한다.

```
DESC table_name;

CREATE TABLE department (
DNO          SMALLINT      PRIMARY KEY,
DNAME        CHAR(30)      NOT NULL,
DEP_LOCATION CHAR(9),
MGR_NO       INTEGER );

isQL> DESC department; ->table_name: 테이블 정보(구조)를 알고 싶은 테이블명
[ TABLESPACE : SYS_TBS_MEM_DATA ]
[ ATTRIBUTE ]
-----
NAME                                TYPE                                IS NULL
-----
DNO                                SMALLINT                            FIXED    NOT NULL
```

DNAME	CHAR(30)	FIXED	NOT NULL
DEP_LOCATION	CHAR(9)	FIXED	
MGR_NO	INTEGER	FIXED	
[ INDEX ]			
-----			
NAME	TYPE	IS UNIQUE	COLUMN
-----			
__SYS_IDX_ID_122	BTREE	UNIQUE	DNO ASC
[ PRIMARY KEY ]			
-----			
DNO			

테이블 이름에 소문자, 특수 문자 또는 공백이 포함된 경우 큰따옴표를 사용해야 한다.

```
iSQL> DESC "table name";
iSQL> DESC "user name"."table name";
```

## 시퀀스 정보 보기

데이터베이스에 존재하는 모든 시퀀스에 대한 정보를 알고 싶으면 아래와 같은 명령을 사용하면 된다.

```
SELECT * FROM seq;

iSQL> CONNECT sys/manager;
Connect success.
iSQL> CREATE USER user1 IDENTIFIED BY user1;
Create success.
iSQL> CONNECT user1/user1;
Connect success.
iSQL> CREATE SEQUENCE seq1 MAXVALUE 100 CYCLE;
Create success.
iSQL> CREATE SEQUENCE seq2;
Create success.
iSQL> CONNECT sys/manager;
Connect success.
iSQL> CREATE SEQUENCE seq2 START WITH 20 INCREMENT BY 30;
Create success.
iSQL> CREATE SEQUENCE seq3 CACHE 40;
Create success.
iSQL> SELECT * FROM seq;
->SYS 계정으로 데이터베이스에 접속한 경우 생성된 모든 sequence의 정보를 출력한다.
```

USER_NAME			
-----			
SEQUENCE_NAME	CURRENT_VALUE	INCREMENT_BY	
-----			
MIN_VALUE	MAX_VALUE	CYCLE	CACHE_SIZE
-----			
SYS			
SEQ2	20		30
1	9223372036854775806	NO	20

```

SYS
SEQ3                                1                                1
1                                9223372036854775806        NO                                40
USER1
SEQ1                                1                                1
1                                100                                YES                                20
USER1
SEQ2                                1                                1
1                                9223372036854775806        NO                                20
4 rows selected.

```

```

iSQL> CONNECT user1/user1;
Connect success.

```

```

iSQL> SELECT * FROM seq;
-> user1이 생성한 모든 sequence 들의 정보를 출력한다.

```

SEQUENCE_NAME	CURRENT_VALUE	INCREMENT_BY
-----	-----	-----
MIN_VALUE	MAX_VALUE	CYCLE
-----	-----	-----
SEQ1	1	1
1	100	YES
20		
SEQ2	1	1
1	9223372036854775806	NO
20		

2 rows selected.

## 트랜잭션 제어

### 트랜잭션 모드 설정

한 번 명령어를 수행할 때마다 자동으로 commit 할 것인지 여부를 결정하는 기능이다.

```

iSQL> AUTOCOMMIT OFF; -> 사용자가 commit 하기 전에는 commit되지 않음
Set autocommit off success.

```

```

iSQL> AUTOCOMMIT ON; -> 명령어를 수행할 때마다 자동으로 commit
Set autocommit on success.

```

## PLANCOMMIT

```

SET PLANCOMMIT ON/OFF;

```

autocommit off (non-autocommit) 모드에서 explain plan이 on (또는 only) 조건일 때, desc, select \* from tab; 또는 select \* from seq; 등과 같은 명령어를 수행했을 때 자동으로 commit 할지를 결정하는 기능이다. 기본값은 OFF 이다.

참고: 기본값이 OFF 이므로 autocommit off 세션에서 explain plan이 on (또는 only) 조건일 때 Altibase는 위의 명령어(desc, select \* from tab; 또는 select \* from seq;)를 자동 commit 하지 않고 오류 메시지를 발생한다.



이 값이 ON이면, iSQL은 이런 명령어들을 실행한 후 commit을 수행해서 에러가 발생하지 않도록 한다.

## 파일 관리

### 작업 결과 저장

iSQL을 통해 작업한 결과를 지정한 파일로 저장하는 기능을 제공한다. 다음과 같이 spool 명령을 이용하면 지정한 book.txt 파일에 작업한 결과가 저장된다.

이 기능을 해제하고 싶으면 SPOOL OFF 명령을 사용한다.

```
iSQL> SPOOL book.txt
Spool start. [book.txt] -> 이후의 모든 명령과 그 결과들이 book.txt 파일에 저장된다. 이
파일은 현재 디렉터리에 생성된다.
iSQL> SPOOL OFF
Spool stop      -> 더 이상 명령과 그 결과들을 파일에 저장하지 않는다.
```

### 스크립트 파일 실행

#### @ 명령어

```
@file_name[.sql]
또는
START file_name[.sql]
```

*file\_name[.sql]*: 수행 될 스크립트 파일, 확장자를 생략하면 iSQL은 기본 스크립트 파일 확장자(.sql)로 간주한다.

iSQL 명령어와 SQL구문들이 저장된 스크립트 파일을 실행하면, 한번에 파일내의 명령어들을 순차적으로 실행한다.

@ 명령어는 START와 같은 기능을 갖는다.

- 스크립트 파일내의 exit 또는 quit 명령어는 iSQL을 종료시킨다.
- 스크립트 파일내에는 일반적으로 SQL문, iSQL 명령어, 또는 Stored Procedure 블록 등이 포함될 수 있다.

다음은 \$ALTIBASE\_HOME/sample/APRE/schema 디렉터리에 있는 스크립트 schema.sql을 현재 디렉터리에서 수행하는 예이다.

```
iSQL> START schema.sql      <- 파일내의 sql 문이 실행된다.
또는
iSQL> @schema.sql
```

스크립트 파일을 명시할 때, 사용자 계정의 Altibase 홈 디렉터리(\$ALTIBASE\_HOME)를 의미하는 물음표("?")를 사용할 수 있다.

다음은 \$ALTIBASE\_HOME/sample/APRE/schema 디렉터리에 있는 스크립트 schema.sql을 다른 디렉터리에서 수행하는 예이다.

```
iSQL> @?/sample/schema.sql
```

물음표("?")는 다음 iSQL 명령어에서도 사용할 수 있다:

edit, save, load, spool, start

스크립트 파일 내에서 주석의 사용은 -- 또는 /\* \*/으로 가능하다.

--는 이 표시 다음부터 그 라인의 끝까지를 주석으로 처리하고 여러 라인을 주석으로 처리할 때는 주석부분을 /\*와 \*/ 사이에 넣으면 된다.

## @@ 명령어

```
@@file_name[.sql]
```

*file\_name[.sql]*: 수행될 내포형 스크립트를 나타낸다. 확장자를 생략하면 iSQL은 기본 명령어 파일 확장자(.sql)로 간주한다.

명시된 스크립트를 수행한다. @@ 명령어는 @ 명령어와 유사한 기능을 갖는다.

이 명령어는 수행될 스크립트와 같은 경로에서 명시된 스크립트를 찾는 기능을 가지고 있기 때문에 내포형 스크립트를 수행하는데 유용하다.

@@ 명령어는 다음과 같은 용도로 쓰일 수 있다.

- 사용자가 임의의 스크립트 파일 내에 @@file\_name.sql을 입력하고 그 스크립트 파일을 실행하면, iSQL은 file\_name.sql을 호출한 스크립트 파일과 동일한 디렉터리에서 file\_name.sql을 찾아서 수행한다. file\_name.sql 은 이를 호출한 스크립트 파일과 같은 디렉터리에 있어야 한다. 만일 그런 파일이 존재하지 않으면, iSQL은 오류 메시지를 보여준다.
- 사용자가 iSQL 프롬프트 상에서 @@file\_name.sql을 입력하여 실행하는 것은 @file\_name.sql을 실행하는 것과 동일하다.
- 스크립트에는 일반적으로 SQL문, iSQL 명령어, 또는 Stored Procedure 블록 등이 포함될 수 있다.
- 스크립트 내에 exit 또는 quit 명령어는 iSQL을 종료시킨다.

다음은 \$ALTIBASE\_HOME 디렉터리에서 a.sql 스크립트 파일을 실행하는 예제로, 이 파일 내에서 schema.sql 스크립트 파일을 실행한다. 이 예제가 에러 없이 제대로 수행되려면, a.sql은 schema.sql파일이 위치하는 \$ALTIBASE\_HOME/sample/APRE/schema 디렉터리에 같이 있어야 한다.

```
iSQL> @sample/APRE/schema/a.sql
```

```
$ cat a.sql
@@schema.sql
```

참고: 이 후의 예제는 위의 스크립트를 실행해서 생성된 테이블들을 (부록 Schema 참고) 가지고 iSQL 환경에서 질의에 따른 결과를 편집한 것이다.

## START 명령에 파라미터 전달

```
START file_name[.sql] [param1 [param2] ...]
@file_name[.sql] [param1 [param2] ...]
@@file_name[.sql] [param1 [param2] ...]
```

[param1 [param2] ...]: 스크립트 파일에 파라미터로 전달할 값

스크립트 파일 내 SQL문의 특정 값을 고정하지 않고 사용자가 수행할 때마다 설정하고자 할 때 치환 변수를 사용한다. START, @ 또는 @@ 명령어로 스크립트 파일 수행 시, 치환 변수 자리에 대체하고자 하는 값을 입력하여 파라미터로 전달할 수 있다.

스크립트 파일 내의 치환 변수는 대체하고자 하는 자리에 &와 숫자를 붙여서 사용하며, 숫자는 순서를 의미한다. 단, 이 기능은 SET DEFINE ON이 설정된 경우에만 동작한다. 자세한 설명은 SET DEFINE을 참조하기 바란다.

예를 들어 아래와 같이 emp.sql 파일에 치환 변수가 사용된 경우

```
SELECT ENO, E_LASTNAME FROM EMPLOYEES
WHERE EMP_JOB = '&1'
AND SALARY > &2;
```

START 명령어 수행 시에 programmer, 2000을 파라미터로 같이 입력하면 &1에 programmer, &2에 2000 값이 대체되어 수행된다. 즉 직업이 programmer이고, 월급이 2000을 초과하는 직원이 조회된다.

```
iSQL> SET DEFINE ON; -- ON으로 설정해야 치환변수를 파라미터 값으로 대체
iSQL> START emp.sql programmer 2000
old  2: WHERE EMP_JOB = '&1'
new  2: WHERE EMP_JOB = 'programmer'
old  3: AND SALARY > &2;
new  3: AND SALARY > 2000;
```

```
ENO          E_LASTNAME
-----
10           Bae
```

iSQL은 치환 변수를 포함하고 있는 라인에 대하여 파라미터 값이 치환되기 전후의 SQL 명령을 함께 출력한다. SET VERIFY OFF를 설정하면 대입 후의 SQL명령은 출력되지 않는다. 치환 변수는 하나의 스크립트에서 여러 번 사용될 수 있으며, 반드시 순서대로 사용하지 않아도 된다.

치환 변수는 아래와 같이 대화식으로도 스크립트의 변수를 파라미터로 치환할 수 있다.

```

START emp.sql
...
Enter value for 1: programmer
old 2: WHERE EMP_JOB = '&1'
new 2: WHERE EMP_JOB = 'programmer'
Enter value for 2: 2000
old 3: AND SALARY > &2;
new 3: AND SALARY > 2000;

```

또한 특정 문자를 치환 변수 뒤에 바로 연결하여 사용하려면, 마침표(.)를 사용하여 치환 변수와 문자를 구분해야 한다.

```

SELECT E_LASTNAME FROM EMPLOYEES WHERE ENO='&1.0';
Enter value for 1: 2
old 1: SELECT E_LASTNAME FROM EMPLOYEES WHERE ENO='&1.0';
new 1: SELECT E_LASTNAME FROM EMPLOYEES WHERE ENO='20';

```

### SET DEFINE {ON|OFF}

치환 변수가 포함된 스크립트 파일을 START, @ 또는 @@ 명령어로 수행 시, 치환 변수를 사용자가 입력한 파라미터 값으로 대체할지 여부를 지정한다.

기본값은 OFF로 사용자가 입력한 파라미터 값으로 치환 변수를 대체하지 않는다. 즉, 치환 변수가 포함된 스크립트 파일 수행 시에는 반드시 ON으로 지정해야 한다.

### SET VERIFY {ON|OFF}

치환 변수가 포함된 스크립트 파일을 START, @ 또는 @@ 명령어로 수행 시, 파라미터 값으로 교체되기 전후의 SQL문을 출력할지 여부를 지정한다.

기본값은 ON으로 전후 SQL문을 출력한다.

```

$cat Param1.sql
SELECT * FROM T1 WHERE I1 = &1;

iSQL> SET DEFINE ON;
iSQL> SHOW VERIFY;
Verify : On
iSQL> START Param1.sql 5;
iSQL> SELECT * FROM T1
WHERE I1 = &1;
old 2: WHERE I1 = &1;
new 2: WHERE I1 = 5;
T1.I1      T1.I2
-----
5          Hyacinth
1 row selected.

iSQL> SET VERIFY OFF;
iSQL> SHOW VERIFY;
Verify : Off
iSQL> START Param1.sql 5;
iSQL> SELECT * FROM T1

```

```
WHERE I1 = &1;
T1.I1      T1.I2
-----
5          Hyacinth
1 row selected.
```

## SQL문의 저장

현재 iSQLbuffer에 있는 명령어 중 가장 최근에 수행한 명령어를 파일로 저장하는 기능이다.

이 파일은 현재 디렉터리에 생성된다.

```
iSQL> SELECT * FROM book;
iSQL> SAVE book.sql; -> book.sql 파일에 'SELECT * FROM book;'가 저장된다.
Save completed.
```

## SQL문의 로드

지정한 파일의 첫 번째 명령어를 iSQL버퍼의 마지막 위치에 로드시키는 기능이다.

```
iSQL> LOAD book.sql
iSQL> SELECT * FROM book;
Load completed.
iSQL> / -> SELECT * FROM book; 문이 실행된 것을 볼 수 있다.
```

## DML문 저장

INSERT, UPDATE, DELETE, MOVE 등의 DML문 실행시 이를 \$ALTIBASE\_HOME/trc/isql\_query.log에 기록한다. 단, DML문 중 SELECT를 실행한 경우에는 로그에 기록되지 않는다.

이 기능을 설정하려면 SET QUERYLOGGING을 ON으로 하고, 해제하려면 OFF하면 된다.

```
iSQL> SET QUERYLOGGING ON; -> 이후의 모든 DML 문이
$ALTIBASE_HOME/trc/isql_query.log에 저장된다.
iSQL> CREATE TABLE T1 ( I1 INTEGER );
Create success.
iSQL> INSERT INTO T1 VALUES ( 1 );
1 row inserted.
iSQL> UPDATE T1 SET I1 = 2;
1 row updated.
iSQL> SELECT * FROM T1;
I1
-----
2
1 row selected.
iSQL> DELETE FROM T1;
1 row deleted.
```

```
iSQL> DROP TABLE T1;
Drop success.
iSQL> EXIT
```

% cat \$ALTIBASE\_HOME/trc/isql\_query.log -> SET QUERYLOGGING ON으로 실행한 후의 DML을 확인할 수 있다.

```
[2009/09/16 10:36:14] [127.0.0.1:25310 SYS] INSERT INTO T1 VALUES ( 1 )
[2009/09/16 10:36:31] [127.0.0.1:25310 SYS] UPDATE T1 SET I1 = 2
[2009/09/16 10:36:37] [127.0.0.1:25310 SYS] DELETE FROM T1
```

## 질의문 편집

### 최근 질의문 편집

iSQL 상에서 파일을 생성하고 편집할 수 있도록 명령어 edit를 제공한다.

인수 없이 ed를 실행하면 가장 최근 실행된 질의문이 iSQL.buf 라는 임시 파일로 생성되며 다음과 같은 화면을 볼 수 있다. (지면을 절약하기 위해 몇 줄만 화면으로 표시한다.)

```
iSQL> SELECT sysdate FROM dual;
SYSDATE
-----
01-JAN-2000
1 row selected.
```

```
iSQL> ed
SELECT sysdate FROM dual;
~
~
~
"iSQL.buf" 1L, 26C
```

### 기존 파일 편집

존재하는 파일을 편집하기 위해서는 iSQL 상에서 ed 실행 시 그 파일 이름을 뒤에 넣으면 된다. 화면이 초기화 되어 있을 때 빈 줄 표시는 ~(tilde) 문자로 표시된다.

```
iSQL> ed myquery.sql
"myquery.sql"
INSERT INTO employee(ENO, E_FIRSTNAME, E_LASTNAME, SEX) VALUES(21, 'MSJUNG',
'F');
INSERT INTO employee(ENO, E_FIRSTNAME, E_LASTNAME, SEX, JOIN_DATE)
VALUES(22, 'Joshua', 'Baldwin', 'M', TO_DATE('2001-11-19 00:00:00', 'YYYY-MM-DD
HH:MI:SS'));
~
~"myquery.sql"
```

## 히스토리 목록에 있는 질의문 편집

히스토리 목록에서 해당 번호를 사용하여 이전에 수행했던 명령을 편집할 수 있다. 즉, 해당 번호의 질의문이 iSQL.buf 임시 파일로 생성되어 편집을 할 수 있으며, 편집 결과는 히스토리의 마지막에 등록되어 가장 마지막 명령을 재수행하는 '/' 으로도 실행이 가능하다.

```
iSQL> h
1 : SELECT * FROM customers;
2 : SELECT * FROM employees;
iSQL> 2ed
or
iSQL> 2 ed
SELECT * FROM employees;
~
~
"iSQL.buf"
```

파일을 편집하기 위해서 명령 줄에 인수 2라는 파일 이름을 넣은 (iSQL> ed 2) 명령어와 구분된다.

편집 후 (employees를 orders로 변경)

```
iSQL> h      <- 현재 iSQL buffer에 있는 히스토리 목록
1 : SELECT * FROM customers;
2 : SELECT * FROM employees;
: SELECT * FROM orders;
    <- 2 ed 명령어에 의해 편집된 질의문이 히스토리 목록에 가장 마지막 명령어로 저장된다.

iSQL> /      <- 가장 최근에 수행한 명령어가 실행된다.
ORDERS.ONO      ORDERS.ORDER_DATE      ORDERS.ENO      ORDERS.CNO
-----
ORDERS.GNO      ORDERS.QTY      ORDERS.ARRIVAL_DATE      ORDERS.PROCESSING
-----
0011290007      2000/11/29 00:00:00  12      7111111431202
A111100002  70      2000/12/02 00:00:00  C
0011290011      2000/11/29 00:00:00  12      7610011000001
E111100001  1000      2000/12/05 00:00:00  D
...
0012310012      2000/12/31 00:00:00  19      7308281201145
C111100001  250      2001/01/03 00:00:00  O
30 rows selected.
```

## 주의사항

파일 이름에 특수 문자 또는 공백이 포함된 경우 큰따옴표를 사용해야 한다.

```
iSQL> SPOOL "file name.txt";
iSQL> START "file name.sql";
iSQL> EDIT "file name.sql";
```

## SELECT 결과 포매팅

SELECT 문에 대한 결과들을 사용자가 보기 좋게 포매팅하는 기능이다.

### SET LINESIZE

SELECT 문 결과 출력시 디스플레이되는 한 라인의 사이즈를 설정한다. 10 에서 200 사이의 값이어야 한다.

```
iSQL> SET LINESIZE 100; --> 한 라인의 디스플레이 크기를 100으로 설정한다.
```

### SET LOBSIZE

CLOB 칼럼을 SELECT 문으로 조회 시 디스플레이 되는 데이터의 길이를 설정한다.

CLOB 칼럼의 데이터를 SELECT 문으로 조회하기 위해서 우선 트랜잭션 모드를 AUTOCOMMIT OFF로 설정해야 한다.

```
CREATE TABLE C1(I1 INTEGER, I2 CLOB);
INSERT INTO C1 VALUES(1, 'A123456789');
INSERT INTO C1 VALUES(2, 'A1234');
INSERT INTO C1 VALUES(3, 'A12345');
INSERT INTO C1 VALUES(4, 'A1234567890123');
```

```
iSQL> autocommit off; --> CLOB 칼럼 조회를 위해 트랜잭션 모드를 OFF로 설정한다.
```

```
Set autocommit off success.
```

```
iSQL> select * from c1;
```

```
C1.I1      C1.I2
```

```
-----
```

```
1      A123456789
```

```
2      A1234
```

```
3      A12345
```

```
4      A1234567890123
```

```
4 rows selected.
```

```
iSQL> set lobsize 10; --> CLOB 칼럼의 데이터를 select 문으로 조회할 때 화면에 나타나는 데이터 길이를 설정한다.
```

```
iSQL> select * from c1;
```

```
C1.I1      C1.I2
```

```
-----
```

```
1          A123456789
```

```
2          A1234
```

```
3          A12345
```

```
4          A123456789
```

```
4 rows selected.
```



## SET LOBOFFSET

CLOB 칼럼을 SELECT 문으로 조회할 때 디스플레이 되는 Clob 데이터의 시작 위치를 설정한다.

CLOB 칼럼의 데이터를 SELECT 문으로 조회하기 위해서 우선 트랜잭션 모드를 AUTOCOMMIT OFF로 설정해야 한다.

```
CREATE TABLE C1(I1 INTEGER, I2 CLOB);
INSERT INTO C1 VALUES(1, 'A123456789');
INSERT INTO C1 VALUES(2, 'A1234');
INSERT INTO C1 VALUES(3, 'A12345');
INSERT INTO C1 VALUES(4, 'A1234567890123');

iSQL> autocommit off;
Set autocommit off success.
iSQL> set loboffset 4; -> CLOB 칼럼의 데이터를 select 문으로 조회할 때 화면에 나타나는
데이터의 시작 위치를 설정한다.
iSQL> select * from c1;
C1.I1      C1.I2
-----
1          456789
2           4
3           45
4          4567890123
4 rows selected.
```

## SET FEEDBACK

SELECT 문 결과 출력시 선택된 결과 건수를 출력한다.

```
SET FEEDBACK ON\|OFF\|n
```

- ON: SELECT문 수행 후 결과 데이터 건수를 출력한다.
- OFF: SELECT문 수행 후 결과 데이터 건수를 출력하지 않는다.
- n: 결과 건수가 n이상일 경우에만 출력한다.

```
iSQL> SET FEEDBACK ON;
iSQL> SELECT * FROM employees WHERE ENO < 3;
ENO      E_LASTNAME      E_FIRSTNAME      EMP_JOB
-----
EMP_TEL      DNO      SALARY      SEX  BIRTH      JOIN_DATE      STATUS
-----
--
1          Moon      Chan-seung      CEO
01195662365      3002      M
2          Davenport      Susan      designer
0113654540      1500      F  721219      18-NOV-2009      H
2 rows selected.
```

## SET PAGESIZE

결과 row들을 몇 개 단위로 보여줄 것인지 결정한다.

```
iSQL> SET PAGESIZE 2;    -> 결과 row를 2개 단위로 보여준다.
```

```
iSQL> SELECT * FROM employees;
```

ENO	E_LASTNAME	E_FIRSTNAME	EMP_JOB
1	Moon	Chan-seung	CEO
01195662365	3002	M	R
2	Davenport	Susan	designer
0113654540	1500	F 721219	18-NOV-2009 H
ENO	E_LASTNAME	E_FIRSTNAME	EMP_JOB
EMP_TEL	DNO	SALARY	SEX BIRTH JOIN_DATE STATUS
3	Kobain	Ken	engineer
0162581369	1001	2000	M 650226 11-JAN-2010 H
4	Foster	Aaron	PL
0182563984	3001	1800	M 820730 H
ENO	E_LASTNAME	E_FIRSTNAME	EMP_JOB
EMP_TEL	DNO	SALARY	SEX BIRTH JOIN_DATE STATUS
5	Ghorbani	Farhad	PL
01145582310	3002	2500	M 20-DEC-2009 H
6	Momoi	Ryu	programmer
0197853222	1002	1700	M 790822 09-SEP-2010 H
.			
.			
.			

20 rows selected.

```
iSQL> SET PAGESIZE 0;    -> 결과 전체를 한 단위로 보여준다.
```

```
iSQL> SELECT * FROM employees;
```

ENO	E_LASTNAME	E_FIRSTNAME	EMP_JOB
1	Moon	Chan-seung	CEO
01195662365	3002	M	R
2	Davenport	Susan	designer
0113654540	1500	F 721219	18-NOV-2009 H
3	Kobain	Ken	engineer
0162581369	1001	2000	M 650226 11-JAN-2010 H
.			
.			
.			

20 rows selected.

## SET HEADING

결과에 헤더를 출력할지 결정한다.

```
iSQL> SET HEADING OFF;    -> 결과에 헤더를 출력하지 않는다.
iSQL> SELECT * FROM employee;
```

```
1          Moon          Chan-seung          CEO
01195662365      3002          M          R
2      Davenport          Susan          designer
0113654540          1500      F  721219  18-NOV-2009  H
3          Kobain          Ken          engineer
0162581369      1001      2000      M  650226  11-JAN-2010  H
.
.
.
20 rows selected.
```

```
iSQL> SET HEADING ON;    -> 결과에 헤더를 출력한다.
iSQL> SELECT * FROM employee;
```

```
ENO          E_LASTNAME          E_FIRSTNAME          EMP_JOB
-----
EMP_TEL          DNO          SALARY          SEX  BIRTH  JOIN_DATE  STATUS
-----
--
1          Moon          Chan-seung          CEO
01195662365      3002          M          R
2      Davenport          Susan          designer
0113654540          1500      F  721219  18-NOV-2009  H
3          Kobain          Ken          engineer
0162581369      1001      2000      M  650226  11-JAN-2010  H
.
.
.
20 rows selected.
```

## SET COLSIZE

SELECT 문 결과 출력시 디스플레이되는 CHAR, VARCHAR 타입 칼럼의 사이즈를 설정하여, 길이가 긴 문자열을 포함하는 칼럼이 존재할 경우 인식을 용이하게 한다.

```
iSQL> CREATE TABLE LOCATION(
ID          INTEGER,
NAME      CHAR(20),
ADDRESS VARCHAR(500),
PHONE      CHAR(20));
Create success.
iSQL> INSERT INTO LOCATION VALUES(1, 'ALTIBASE', 'Inyoung Bldg, 5fl 44-11 Youido-
dong Youngdungpo-qu seoul, 150-890. Korea', '82-2-769-7500');
1 row inserted.
```

아래는 CHAR 또는 VARCHAR 타입 칼럼의 디스플레이 사이즈를 7로 설정하여 조회하는 예제이다.

```
iSQL> SET COLSIZE 7;
iSQL> SELECT ID,NAME,ADDRESS,PHONE FROM LOCATION;
ID          NAME      ADDRESS  PHONE
-----
1           ALTIBAS  10Fl.,   82-2-20
              E      Daerung  82-1000
              post-to
              wer II,
              Guro-d
              ong, Gu
              ro-qu,
              Seoul 1
              52-790.
              Korea

1 row selected.
```

## SET NUM[WIDTH]

NUMERIC, DECIMAL, NUMBER, FLOAT 타입의 SELECT 결과를 표시할 자리수를 설정한다. 유효숫자의 개수가 많은 데이터는 이 값을 크게 설정하면 읽기가 쉽다.

아래는 NUMWIDTH를 30으로 설정한 후 NUMERIC, DECIMAL, NUMBER, FLOAT 칼럼을 조회하는 예제이다.

```
iSQL> CREATE TABLE t1
(
  c_numeric NUMERIC(38, 0),
  c_decimal DECIMAL(38, 0),
  c_number NUMBER(38, 0),
  c_float FLOAT(38)
);
Create success.
iSQL> INSERT INTO t1 VALUES(12345678901234567890, 12345678901234567890,
12345678901234567890, 12345678901234567890);
1 row inserted.
iSQL> SET NUMWIDTH 30
iSQL> SELECT c_numeric, c_decimal, c_number, c_float FROM t1;
C_NUMERIC C_DECIMAL
-----
C_NUMBER C_FLOAT
-----
12345678901234567890 12345678901234567890
12345678901234567890 12345678901234567890
1 row selected.
```

## SET NUMF[ORMAT]

### 구문

```
SET NUMF[ORMAT] format;
```

NUMERIC, DECIMAL, NUMBER, FLOAT 타입의 SELECT 결과를 표시할 수 있는 형식을 설정한다. 이 설정은 SET NUMWIDTH 설정보다 우선한다.

*format*에 설정할 수 있는 형식은 "*General Reference*> 자료형> 숫자형 데이터 타입> 숫자형 데이터 형식"을 참고한다.

아래는 지수 형식으로 조회하는 예제이다.

```
iSQL> create table t1(i1 float(30));
Create success.
iSQL> insert into t1 values (123456789012);
1 row inserted.
iSQL> SET NUMFORMAT 9.99EEEE
iSQL> select * from t1;
T1.I1
-----
      1.23E+11
1 rows selected.
```

## CL[EAR] COL[UMNMS]

COLUMN 명령어로 설정된 모든 칼럼의 표시 형식을 해제한다.

### 구문

```
CL[EAR] COL[UMNS]
```

## COLUMN

SELECT 대상(target)이 되는 칼럼의 표시 형식을 설정하거나 확인할 수 있다. 아래의 데이터 타입일 경우에만 설정이 적용된다.

- 문자형 데이터 타입의 길이
- 숫자형 데이터 타입의 표시 형식

### 구문

```
COL[UMN] [{column | expr} [option]]
```

*column* 또는 *expr*은 SELECT 대상이 되는 칼럼 또는 식을 가리키며, select 문에 사용한 것과 동일하게 기술해야 한다. COL[UMN] [{*column* | *expr*}] 명령어만 사용하면 설정한 모든 칼럼 또는 지정한 칼럼에 설정된 형식을 확인할 수 있다.

*option*에서 설정할 수 있는 기능은 아래와 같다.

옵션	설명
CLE[AR]	지정된 칼럼의 설정을 해제
FOR[MAT] format	칼럼의 표시 형식을 설정한다. 문자형 칼럼: CHAR, VARCHAR 타입의 표시 길이 설정. format은 A10과 같은 텍스트 상수로 지정할 수 있다. SET COLSIZE 설정보다 우선한다. 숫자형 칼럼 : NUMBER, DECIMAL, FLOAT, NUMERIC 타입의 표시 형식 설정 format에 설정할 수 있는 형식은 "General Reference> 자료형 > 숫자형 데이터 타입 > 숫자형 데이터 형식"을 참고한다. SET NUMFORMAT 설정보다 우선한다.
ON OFF	설정된 표시 형식의 적용 여부 OFF: 칼럼에 대한 설정은 그대로 둔 채, 출력에는 적용하지 않는다. ON : 설정을 적용한다.

## 설명

SELECT 구문의 대상 칼럼에 대한 표시 형식을 설정할 수 있다. 동일한 칼럼을 여러 개의 표시 형식으로 설정하면 마지막 형식이 적용된다.

설정된 표시 형식을 적용하지 않으려면 CLEAR 또는 OFF 옵션을 사용할 수 있다. CLEAR 옵션은 설정을 완전히 해제하는 것이며, OFF는 출력만 적용되지 않는 차이가 있다.

## 예제

아래는 VARCHAR(60)의 address 칼럼의 길이를 20으로 표시하려는 예를 보여준다.

```
isQL> @schema.sql
isQL> COLUMN address FORMAT A20
isQL> select cno, address from customers;
CNO                ADDRESS
-----
1                  2100 Exposition Boul
                  evard Los Angeles US
                  A
...
```

설정을 삭제하려면 아래와 같이 실행한다.

```
isQL> COLUMN address CLE
```

## 출력 옵션

### 수행시간 출력

SQL 문을 실행하는데 걸린 시간을 알려주는 기능이다.

```
isQL> SET TIMING ON;      -> 명령 실행 후 마지막 라인에 실행시간을 출력한다.
isQL> SELECT * FROM employees;
ENO                E_LASTNAME                E_FIRSTNAME                EMP_JOB
```

```

-----
EMP_TEL          DNO          SALARY      SEX  BIRTH  JOIN_DATE  STATUS
-----
--
1              Moon              Chan-seung      CEO
01195662365      3002              M              R
2              Davenport      Susan              designer
0113654540              1500      F  721219  18-NOV-2009  H
.
.
.
20 rows selected.
elapsed time : 0.01
iSQL> SET TIMING OFF;    -> 실행시간을 출력하지 않는다.

```

## 수행시간 단위 설정

SQL문의 쿼리 수행 시간 단위를 설정하는 기능이다. 설정할 수 있는 단위는 다음과 같다.

- 초
- 밀리초
- 마이크로초
- 나노초

```

iSQL> SET TIMING ON
iSQL> SET TIMESCALE SEC;
iSQL> SELECT * FROM employees;

```

```

ENO          E_LASTNAME          E_FIRSTNAME          EMP_JOB
-----

```

```

EMP_TEL          DNO          SALARY      SEX  BIRTH  JOIN_DATE  STATUS
-----

```

```

--
1              Moon              Chan-seung      CEO
01195662365      3002              M              R
...

```

```

20 rows selected.
elapsed time : 0.00

```

```

iSQL> SET TIMESCALE MILSEC;
iSQL> SELECT * FROM employee;

```

```

ENO          E_LASTNAME          E_FIRSTNAME          EMP_JOB
-----

```

```

EMP_TEL          DNO          SALARY      SEX  BIRTH  JOIN_DATE  STATUS
-----

```

```

--
1              Moon              Chan-seung      CEO
01195662365      3002              M              R
...

```

```

20 rows selected.
elapsed time : 0.72

```

```
iSQL> SET TIMESCALE MICSEC;
iSQL> SELECT * FROM employee;
ENO          E_LASTNAME          E_FIRSTNAME          EMP_JOB
-----
EMP_TEL      DNO          SALARY          SEX  BIRTH  JOIN_DATE  STATUS
-----
--
1            Moon            Chan-seung          CEO
01195662365  3002              M                  R
...
20 rows selected.
elapsed time : 966.00

iSQL> SET TIMESCALE NANSEC;
iSQL> SELECT * FROM employee;
NO          E_LASTNAME          E_FIRSTNAME          EMP_JOB
-----
EMP_TEL      DNO          SALARY          SEX  BIRTH  JOIN_DATE  STATUS
-----
--
1            Moon            Chan-seung          CEO
01195662365  3002              M                  R
...
20 rows selected.
elapsed time : 681000.00
```

## 외래키 출력

DESC 명령어를 사용하여 테이블 구조를 볼 때 외래 키에 대한 정보를 보여주는 기능이다.

```

iSQL> SET FOREIGNKEYS ON;      -> 외래 키에 대한 정보를 출력한다.
iSQL> DESC employees;
[ TABLESPACE : SYS_TBS_MEM_DATA ]
[ ATTRIBUTE ]

```

NAME	TYPE		IS NULL
ENO	INTEGER	FIXED	NOT NULL
E_LASTNAME	CHAR(20)	FIXED	NOT NULL
E_FIRSTNAME	CHAR(20)	FIXED	NOT NULL
EMP_JOB	VARCHAR(15)	FIXED	
EMP_TEL	CHAR(15)	FIXED	
DNO	SMALLINT	FIXED	
SALARY	NUMERIC(10, 2)	FIXED	
SEX	CHAR(1)	FIXED	
BIRTH	CHAR(6)	FIXED	
JOIN_DATE	DATE	FIXED	
STATUS	CHAR(1)	FIXED	

```

[ INDEX ]

```

NAME	TYPE	IS UNIQUE	COLUMN
------	------	-----------	--------



```

-----
__SYS_IDX_ID_238          BTREE      UNIQUE          ENO ASC
EMP_IDX1                  BTREE          DNO ASC
[ PRIMARY KEY ]
-----

```

```

ENO

```

```

[ FOREIGN KEYS ]
-----

```

```

iSQL> SET FOREIGNKEYS OFF;  -> 외래 키에 대한 정보를 출력하지 않는다.

```

```

iSQL> DESC employees;

```

```

[ ATTRIBUTE ]
-----

```

NAME	TYPE		IS NULL
ENO	INTEGER	FIXED	NOT NULL
E_LASTNAME	CHAR(20)	FIXED	NOT NULL
E_FIRSTNAME	CHAR(20)	FIXED	NOT NULL
EMP_JOB	VARCHAR(15)	FIXED	
EMP_TEL	CHAR(15)	FIXED	
DNO	SMALLINT	FIXED	
SALARY	NUMERIC(10, 2)	FIXED	
SEX	CHAR(1)	FIXED	
BIRTH	CHAR(6)	FIXED	
JOIN_DATE	DATE	FIXED	
STATUS	CHAR(1)	FIXED	

```

[ INDEX ]
-----

```

NAME	TYPE	IS UNIQUE	COLUMN
__SYS_IDX_ID_238	BTREE	UNIQUE	ENO ASC
EMP_IDX1	BTREE		DNO ASC

```

[ PRIMARY KEY ]
-----

```

```

ENO

```

## Check 제약조건 출력

DESC 명령어를 사용하여 테이블 구조를 볼 때 Check 제약조건에 대한 정보를 보여주는 기능이다.

```

iSQL> SET CHKCONSTRAINTS ON;          -> Check Constraint에 대한 정보를 출력한다.

```

```

iSQL> DESC employees;

```

```

[ TABLESPACE : SYS_TBS_MEM_DATA ]

```

```

[ ATTRIBUTE ]
-----

```

NAME	TYPE		IS NULL
ENO	INTEGER	FIXED	NOT NULL
E_LASTNAME	CHAR(20)	FIXED	NOT NULL
E_FIRSTNAME	CHAR(20)	FIXED	NOT NULL
EMP_JOB	VARCHAR(15)	FIXED	

EMP_TEL	CHAR(15)	FIXED
DNO	SMALLINT	FIXED
SALARY	NUMERIC(10, 2)	FIXED
SEX	CHAR(1)	FIXED
BIRTH	CHAR(6)	FIXED
JOIN_DATE	DATE	FIXED
STATUS	CHAR(1)	FIXED
[ INDEX ]		

NAME	TYPE	IS UNIQUE	COLUMN
__SYS_IDX_ID_238	BTREE	UNIQUE	ENO ASC
EMP_IDX1	BTREE		DNO ASC
[ PRIMARY KEY ]			

ENO

[ CHECK CONSTRAINTS ]

NAME : EMP\_CHECK\_SEX1  
CONDITION : SEX in ('M', 'F')

iSQL> SET CHKCONSTRAINTS OFF;      -> Check Constraint에 대한 정보를 출력하지 않는다.

iSQL> DESC employees;  
[ TABLESPACE : SYS\_TBS\_MEM\_DATA ]  
[ ATTRIBUTE ]

NAME	TYPE		IS NULL
ENO	INTEGER	FIXED	NOT NULL
E_LASTNAME	CHAR(20)	FIXED	NOT NULL
E_FIRSTNAME	CHAR(20)	FIXED	NOT NULL
EMP_JOB	VARCHAR(15)	FIXED	
EMP_TEL	CHAR(15)	FIXED	
DNO	SMALLINT	FIXED	
SALARY	NUMERIC(10, 2)	FIXED	
SEX	CHAR(1)	FIXED	
BIRTH	CHAR(6)	FIXED	
JOIN_DATE	DATE	FIXED	
STATUS	CHAR(1)	FIXED	
[ INDEX ]			

NAME	TYPE	IS UNIQUE	COLUMN
__SYS_IDX_ID_238	BTREE	UNIQUE	ENO ASC
EMP_IDX1	BTREE		DNO ASC
[ PRIMARY KEY ]			

ENO

## 파티션 출력

DESC 명령어를 사용하여 테이블 구조를 볼 때 파티션에 대한 정보도 함께 보여주는 기능이다.

```
iSQL> create table t1_range(
c1 integer,
c2 integer,
c3 varchar(4))
PARTITION BY RANGE(c3)
(
PARTITION P_2000 VALUES LESS THAN ('2001') TABLESPACE sys_tbs_disk_data,
PARTITION P_2001 VALUES LESS THAN ('2002') TABLESPACE sys_tbs_mem_data,
PARTITION P_DEFAULT VALUES DEFAULT
) tablespace SYS_TBS_DISK_DATA;
```

iSQL> SET PARTITIONS ON; -> 파티션에 대한 정보를 출력한다.

iSQL> DESC t1\_range

[ TABLESPACE : SYS\_TBS\_DISK\_DATA ]

[ ATTRIBUTE ]

NAME	TYPE	IS NULL
------	------	---------

C1	INTEGER	
----	---------	--

C2	INTEGER	
----	---------	--

C3	VARCHAR(4)	
----	------------	--

T1\_RANGE has no index

T1\_RANGE has no primary key

[ PARTITIONS ]

Method: Range

Key column(s)

NAME

C3

Values

PARTITION NAME	MIN VALUE	MAX VALUE
----------------	-----------	-----------

P_2000		'2001'
--------	--	--------

P_2001	'2001'	'2002'
--------	--------	--------

P_DEFAULT	'2002'	
-----------	--------	--

Tablespace

PARTITION NAME	TABLESPACE NAME
----------------	-----------------

P_2000	SYS_TBS_DISK_DATA
--------	-------------------

P_2001	SYS_TBS_MEM_DATA
--------	------------------

P_DEFAULT	SYS_TBS_DISK_DATA
-----------	-------------------

iSQL> SET PARTITIONS OFF; -> 파티션에 대한 정보를 출력하지 않는다.

iSQL> DESC t1\_range

[ TABLESPACE : SYS\_TBS\_DISK\_DATA ]

[ ATTRIBUTE ]

NAME	TYPE	IS NULL
-----		
C1	INTEGER	
C2	INTEGER	
C3	VARCHAR(4)	
T1_RANGE has no index		
T1_RANGE has no primary key		

## 스크립트 파일 실행결과 및 명령어 출력

SET TERM과 SET ECHO 명령어는 스크립트 파일 실행의 결과 및 명령어를 화면상에 보여줄지를 결정한다. 그러나 iSQL 프롬프트 상에서 질의를 직접 입력하는 경우 (예: iSQL> select \* from t1;)에는 TERM과 ECHO 설정이 출력에 영향을 미치지 않는다.

스크립트 파일의 실행 결과는 출력(TERM ON)을 기본으로 한다. SET TERM 명령어를 사용해서 TERM 값을 변경하면 ECHO 값도 자동으로 TERM 값과 동일하게 변경된다. 예를 들어, TERM 값을 OFF로 설정하면 iSQL 상에서 스크립트 파일이 실행될 때 스크립트 파일 내의 명령어와 실행 결과가 화면에 출력되지 않는다.

TERM ON 상태에서 ECHO를 OFF로 설정하면 명령어는 출력되지 않고 실행 결과만 화면에 출력된다. 그리고 TERM을 OFF로 설정했더라도 ECHO를 ON으로 설정하면 스크립트 내의 실행된 명령어를 출력할 수 있다.

다음은 스크립트 파일 실행 결과를 출력하는 예제이다.

```
iSQL> SET TERM ON;          -> 스크립트 실행 결과를 출력한다.
iSQL> @schema.sql
iSQL> ALTER SESSION SET AUTOCOMMIT = TRUE;      ->결과 시작
Alter success.
iSQL> DROP TABLE ORDERS;
Drop success.
elapsed time : 0.00
iSQL> DROP TABLE EMPLOYEES;
Drop success.
elapsed time : 0.00
.
.
.
iSQL> CREATE INDEX ODR_IDX3 ON ORDERS (GNO ASC);
Create success.
elapsed time : 0.00      -> 결과 끝
```

아래는 TERM OFF로 설정했더라도, ECHO를 ON 상태로 변경하면 @으로 실행된 스크립트 내의 명령어를 출력하는 예제이다.

```
iSQL> SET TERM OFF;          -> 스크립트 실행 결과를 출력하지 않는다.
iSQL> @schema.sql
iSQL> SELECT eno, e_firstname, e_lastname FROM employees;
      -> 질의를 직접 입력하는 경우 결과는 출력된다.
ENO          E_FIRSTNAME          E_LASTNAME
-----
1            Chan-seung           Moon
2            Susan                Davenport
```

```

3          Ken          Kobain
4          Aaron         Foster
5          Farhad        Ghorbani
.
.
.
iSQL> SET ECHO ON;  -> @으로 실행된 스크립트 내의 명령어만 출력한다.
iSQL> @schema.sql
ALTER SESSION SET AUTOCOMMIT = TRUE;
DROP TABLE ORDERS;
DROP TABLE EMPLOYEES;
.
.
.
iSQL> CREATE INDEX ODR_IDX3 ON ORDERS (GNO ASC);
Create success.
elapsed time : 0.00  -> 결과 끝

```

## 실행 계획 출력

SQL 튜닝을 위하여 iSQL상에서 실행 계획 (Explain Plan)을 출력하는 기능이다. SQL 문의 실행 계획은 SELECT, INSERT, UPDATE, DELETE 등의 DML 문에 대해서 확인이 가능하다.

이를 위해서 SELECT 등의 구문을 수행하기 전에 다음 명령을 수행하여야 한다.

```
ALTER SESSION SET EXPLAIN PLAN = option;
```

여기서 option은 ON, OFF, ONLY의 세 가지 설정이 있으며, 기본 설정값은 OFF이다.

- ON: SELECT문 실행 후 결과 레코드와 함께 Execution Plan 을 보여준다.
- ONLY: SELECT문에 대해 Prepare 과정만 수행한 후 Execution 과정을 수행하지 않고 실행 계획만 보여준다. 주 언어 변수 바인딩이 존재하는 SELECT 문 또는 실행 수행 시간이 오래 걸리는 질의에 대해 단순히 실행 계획만 확인할 경우 이 기능을 사용한다.
- OFF: SELECT문 실행 후 결과 레코드만 보여준다.

사용자가 기술한 WHERE절에 존재하는 조건들의 처리 방법 등의 보다 자세한 정보가 필요한 경우는 다음 명령을 사용한다.

```
ALTER SYSTEM SET TRCLOG_DETAIL_PREDICATE = 1;
```

위의 구문처럼 해당 프로퍼티를 1로 설정하여 ON시키면, 실행 계획 정보에 WHERE절의 조건들이 FIXED KEY RANGE, VARIABLE KEY RANGE, FILTER 등으로 자세하게 분류되어 표시된다. 따라서 WHERE절을 복잡하게 사용한 경우 어떤 술어들이 인덱스 스캔을 통해 수행되는지 확인할 수 있다. 단, 특정 최적화 기법에 의해 질의가 변경된 경우는 이러한 정보가 출력되지 않을 수 있다.

다음은 해당 SQL문을 사용한 출력 예이다.

- TRCLOG\_DETAIL\_PREDICATE을 설정하고 EXPLAIN PLAN = ON으로 한 경우

```

isQL> alter system set trclog_detail_predicate = 1;
Alter success.
isQL> alter session set explain plan = on;
Alter success.
isQL> SELECT eno, e_lastname, e_firstname FROM employees WHERE eno = 1;
ENO          E_LASTNAME          E_FIRSTNAME
-----
1            Moon              Chan-seung
1 row selected.
-----
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 48 )
  SCAN ( TABLE: EMPLOYEES, INDEX: __SYS_IDX_ID_238, ACCESS: 1, SELF_ID: 2 )
    [ FIXED KEY ]
  AND
    OR
    ENO = 1
-----

```

- TRCLOG\_DETAIL\_PREDICATE을 설정하지 않고, EXPLAIN PLAN = ON으로 한 경우

```

isQL> ALTER SYSTEM SET TRCLOG_DETAIL_PREDICATE = 0;
Alter success.
isQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
isQL> SELECT eno, e_lastname, e_firstname FROM employees WHERE eno = 1;
ENO          E_LASTNAME          E_FIRSTNAME
-----
1            Moon              Chan-seung
1 row selected.
-----
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 48 )
  SCAN ( TABLE: EMPLOYEES, INDEX: __SYS_IDX_ID_238, ACCESS: 1, SELF_ID: 2 )
-----

```

- TRCLOG\_DETAIL\_PREDICATE을 설정하지 않고, EXPLAIN PLAN = ONLY로 한 경우

```

isQL> ALTER SYSTEM SET TRCLOG_DETAIL_PREDICATE = 0;
Alter success.
isQL> ALTER SESSION SET EXPLAIN PLAN = ONLY;
Alter success.
isQL> SELECT eno, e_lastname, e_firstname FROM employees WHERE eno = 1;
ENO          E_LASTNAME          E_FIRSTNAME
-----
No rows selected.
-----
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 48 )
  SCAN ( TABLE: EMPLOYEES, INDEX: __SYS_IDX_ID_238, ACCESS: ??, SELF_ID: 2 )
-----

```

EXPLAIN PLAN = ONLY인 경우 질의 실행 없이 실행 계획만 생성하므로 ACCESS 항목과 같이 실제 실행 후 그 값이 결정되는 항목들은 물음표 ("??")로 표시된다.

## 결과 출력 방향 설정

iSQL에서 SELECT 구문으로 조회할 경우, 결과를 세로로 보여줄 것인가 가로로 보여줄 것인가를 선택할 수 있다.

조회 결과가 행이 적고, 열이 많을 경우에 적합한 출력이다. 일반적으로 이러한 경우에 가로로 출력되면 해당하는 열을 맞춰 값을 보기가 어렵다. 하지만 출력 결과의 방향을 세로로 설정하면 보기가 편리하다.

```
iSQL>SET VERTICAL ON;          -> 출력 결과 모드를 세로로 설정한다.
iSQL> SELECT * FROM employees WHERE eno = 2;
ENO          : 2
E_LASTNAME   : Davenport
E_FIRSTNAME  : Susan
EMP_JOB      : designer
EMP_TEL      : 0113654540
DNO          :
SALARY       : 1500
SEX          : F
BIRTH        : 721219
JOIN_DATE    : 18-NOV-2009
STATUS       : H

1 row selected.
```

## iSQL 화면 설정 보기

다음은 현재 세션에서 iSQL 화면 설정 값을 보는 예를 보여준다.

```
iSQL> SHOW USER   -> 현재 세션에 접속한 사용자
User : SYS
iSQL> SHOW COLSIZE
ColSize : 0
iSQL> SHOW LOBOFFSET
LobOffset: 0
iSQL> SHOW LINESIZE
Linesize : 100
iSQL> SHOW LOBSIZE
LobSize : 80
iSQL> SHOW NUMWIDTH
NumWidth : 11
iSQL> SHOW PAGESIZE
Pagesize : 0
iSQL> SHOW TIMESCALE
TimeScale : Second
iSQL> SHOW HEADING
Heading : On
iSQL> SHOW TIMING
Timing : Off
iSQL> SHOW VERTICAL
Vertical : Off
iSQL> SHOW CHKCONSTRAINTS
ChkConstraints : Off
```

```
iSQL> SHOW FOREIGNKEYS
ForeignKeys : Off
iSQL> SHOW PLANCOMMIT
PlanCommit : Off
iSQL> SHOW QUERYLOGGING
QueryLogging : Off
iSQL> SHOW TERM
Term : On
iSQL> SHOW ECHO
Echo : OFF
iSQL> SHOW FEEDBACK
Feedback : 1
iSQL> SHOW ALL
User      : SYS
ColSize   : 0
LobOffset : 0
LineSize  : 80
LobSize   : 80
NumWidth  : 11
PageSize  : 0
TimeScale : Second
Heading   : On
Timing    : Off
Vertical  : off
ChkConstraints : Off
ForeignKeys : Off
Partitions : off
PlanCommit : Off
QueryLogging : Off
Term : On
Echo : Off
Feedback : 1
Fullname : off
Sqlprompt : "iSQL> "
Define : Off
```

## 호스트 변수

호스트 변수를 선언하여 사용할 수 있다. 호스트 변수는 프로시저나 함수 실행 시 유용하다.

### 호스트 변수 선언하기

#### 구문

```
VAR[TABLE] var_name[INPUT|OUTPUT|INOUTPUT] var_type
```

INPUT 또는 OUTPUT 또는 INOUTPUT을 명시하지 않으면 기본값은 자동으로 부여된다.



## 타입

변수 선언 시 사용할 수 있는 타입은 다음과 같다.

```
INTEGER, BYTE(n), NIBBLE(n),
NUMBER, NUMBER(n), NUMBER(n,m),
NUMERIC, NUMERIC(n), NUMERIC(n,m),
CHAR(n), VARCHAR(n), NCHAR(n), NVARCHAR(n), DATE
DECIMAL, DECIMAL(n), DECIMAL(n,m),
FLOAT, FLOAT(n), DOUBLE, REAL
BIGINT, SMALLINT
```

## 예제

아래는 변수를 선언하는 예를 보여준다.

```
iSQL> VAR p1 INTEGER
iSQL> VAR p2 CHAR(10)
iSQL> VAR v_double DOUBLE
iSQL> VAR v_real REAL
```

## 호스트 변수에 값 할당하기

### 구문

```
EXEC[UTE] :var_name := value;
```

## 예제

아래는 변수에 값을 할당하는 예를 보여준다.

```
iSQL> EXECUTE :p1 := 100;
Execute success
iSQL> EXEC :p2 := 'abc';
Execute success
```

## 호스트 변수 보기

### 구문

```
PRINT VAR[IABLE]
```

선언된 모든 변수를 보여준다.

```
PRINT var_name
```

var\_name의 타입과 값을 보여준다.

## 예제

다음은 선언된 모든 변수값을 보여준다.

```
iSQL> PRINT VAR
[ HOST VARIABLE ]

-----
NAME                TYPE                VALUE
-----
P1                  INTEGER             100
P2                  CHAR(10)             abc
V_REAL              REAL
V_DOUBLE            DOUBLE
iSQL> PRINT p2  -> 변수 p2에 관한 정보만 출력한다.
NAME                TYPE                VALUE
-----
P2                  CHAR ( 10 )          abc
```

## PREPARE SQL문 수행

### Prepared SQL문 수행과 Direct SQL문 수행의 차이

iSQL상에서 SQL문을 수행하면 기본적으로 Direct Execution 방법으로 수행된다. Direct Execution 이란 질의에 대한 구문 분석, 정당성 검사, 최적화 및 수행을 한번에 수행하는 것을 의미한다.

Prepared Execution 방법은 prepare를 할 때 질의에 대한 구문 분석, 정당성 검사, 최적화까지만 수행하여 질의의 실행 계획을 미리 수립한다. 그 후 클라이언트로부터 실행을 요청받으면, 수행을 하는 것이다. ODBC를 사용한 응용프로그램 작성의 경우 Prepared Execution 방법이 일반적으로 사용되며, 호스트 변수 바인딩이 된 SQL문에 대한 반복적인 수행이 필요한 경우에 속도의 이점을 볼 수 있다.

두 가지 수행 방법의 차이는 iSQL 상에서 변수 사용 여부의 차이가 존재하지만, 속도의 차이는 없다. 단, Prepared Execution 방법으로 수행하는 경우, 출력되는 그래프 정보와 실행 계획 정보가 다를 수 있다. 그래프 정보는 최적화까지의 중간 계획을 나타내고, 실행 계획은 변수에 실제 값이 적용된 이후의 계획을 나타내기 때문이다.

### Prepared SQL문

#### 구문

```
PREPARE SQL_statement;
```

## 예제

다음은 PREPARE 명령어를 사용한 SQL문 수행 예제이다.

```
iSQL> VAR t1 INTEGER;
iSQL> EXEC :t1 := 1;
Execute success.
iSQL> PREPARE SELECT eno, e_firstname, e_lastname, sex
FROM employees WHERE eno=:t1;
ENO          E_FIRSTNAME          E_LASTNAME          SEX
-----
1            Chan-seung            Moon                M
1 row selected.
```

## 프로시저 생성과 실행 및 삭제

### 프로시저 생성

프로시저를 생성하는 기능을 제공한다. 프로시저 생성시 반드시 아래의 구문으로 끝나야 한다.

```
END;
/
```

생성된 프로시저는 sys\_procedures\_ 메타 테이블을 참조하여 확인할 수 있다.

### 프로시저 실행

프로시저를 실행하는 기능을 제공한다. 프로시저를 실행함으로써 다양한 쿼리를 한꺼번에 수행할 수 있다. 실행할 프로시저에 파라미터가 있는 경우 반드시 프로시저 실행전에 파라미터 개수만큼 변수가 선언되어 있어야 한다.

### 예제1

다음은 INSERT 문을 수행하는 프로시저 emp\_proc를 생성하는 예를 보여준다. (IN 파라미터 이용)

```
iSQL> CREATE OR REPLACE PROCEDURE emp_proc(p1 IN INTEGER, p2 IN CHAR(20), p3 IN
CHAR(20), p4 IN CHAR(1))
AS
BEGIN
INSERT INTO employees(enno, e_firstname, e_lastname, sex)
VALUES(p1, p2, p3, p4);
END;
/
Create success.
iSQL> SELECT * FROM system_.sys_procedures_ order by created desc limit 1;
USER_ID      PROC_OID
-----
PROC_NAME          OBJECT_TYPE STATUS
-----
PARA_NUM      RETURN_DATA_TYPE RETURN_LANG_ID RETURN_SIZE
-----
RETURN_PRECISION RETURN_SCALE PARSE_NO      PARSE_LEN      CREATED
-----
```

```

LAST_DDL_TIME
-----
2          3208680
EMP_PROC          0          0
4
          2          192          29-FEB-2012
29-FEB-2012
1 row selected.

```

아래는 emp\_proc를 실행하는 예를 보여준다.

```

iSQL> VAR eno INTEGER
iSQL> VAR first_name CHAR(20)
iSQL> VAR last_name CHAR(20)
iSQL> VAR sex CHAR(1)
iSQL> EXECUTE :eno := 21;
Execute success.
iSQL> EXECUTE :first_name := 'Joel';
Execute success.
iSQL> EXECUTE :last_name := 'Johnson';
Execute success.
iSQL> EXECUTE :sex := 'M';
Execute success.
iSQL> EXECUTE emp_proc(:eno, :first_name, :last_name, :sex);
Execute success.
iSQL> SELECT eno, e_firstname, e_lastname, sex FROM employees WHERE eno = 21;
ENO          E_FIRSTNAME          E_LASTNAME          SEX
-----
21          Joel          Johnson          M
1 row selected.

```

## 예제2

다음은 SELECT 문을 수행하는 프로시저 outProc를 생성하는 예를 보여준다.

```

iSQL> CREATE TABLE outTbl(i1 INTEGER, i2 INTEGER);
Create success.
iSQL> INSERT INTO outTbl VALUES(1,1);
1 row inserted.
iSQL> /
1 row inserted.
iSQL> /
1 row inserted.
iSQL> /
1 row inserted.
iSQL> /
1 row inserted.
iSQL> SELECT * FROM outTbl;
OUTTBL.I1    OUTTBL.I2
-----
1            1
1            1
1            1

```

```

1          1
1          1
5 rows selected.
iSQL> CREATE OR REPLACE PROCEDURE outProc(a1 OUT INTEGER, a2 IN OUT INTEGER)
AS
BEGIN
    SELECT COUNT(*) INTO a1 FROM outTbl WHERE i2 = a2;
END;
/
Create success.

```

아래는 outProc를 실행하는 예를 보여준다.

```

iSQL> VAR t3 INTEGER
iSQL> VAR t4 INTEGER
iSQL> EXEC :t4 := 1;
Execute success.
iSQL> EXEC outProc (:t3, :t4);
Execute success.
iSQL> PRINT t3;

```

NAME	TYPE	VALUE
T3	INTEGER	5

### 예제3

다음은 프로시저 outProc1을 생성하는 예를 보여준다.

```

iSQL> CREATE OR REPLACE PROCEDURE outProc1( p1 INTEGER, p2 IN OUT INTEGER, p3 OUT
INTEGER)
AS
BEGIN
    p2 := p1;
    p3 := p1 + 100;
END;
/
Create success.
iSQL> VAR v1 INTEGER
iSQL> VAR v2 INTEGER
iSQL> VAR v3 INTEGER
iSQL> EXEC :v1 := 3;
Execute success.
iSQL> EXEC outProc1(:v1, :v2, :v3);
Execute success.
iSQL> PRINT VAR;
[ HOST VARIABLE ]

```

NAME	TYPE	VALUE
..		
V1	INTEGER	3
V2	INTEGER	3
V3	INTEGER	103

#### 예제4

다음은 SELECT 문을 수행하는 프로시저 inoutProc를 생성하는 예를 보여준다.

```

iSQL> CREATE TABLE inoutTbl(i1 INTEGER);
Create success.
iSQL> INSERT INTO inoutTbl VALUES(1);
1 row inserted.
iSQL> /
1 row inserted.
iSQL> /
1 row inserted.
iSQL> SELECT * FROM inoutTbl;
INOUTTBL.I1
-----
1
1
1
3 rows selected.
iSQL> CREATE OR REPLACE PROCEDURE inoutProc (a1 IN OUT INTEGER)
AS
BEGIN
    SELECT COUNT(*) INTO a1 FROM inoutTbl WHERE i1 = a1;
END;
/
Create success.
iSQL> VAR t3 INTEGER
iSQL> EXEC :t3 := 1;
Execute success.
iSQL> EXEC inoutProc(:t3);
Execute success.
iSQL> PRINT t3;
NAME                TYPE                VALUE
-----
T3                   INTEGER             3

```

#### 예제5

다음은 프로시저 inoutProc1을 생성하는 예를 보여준다.

```

iSQL> CREATE OR REPLACE PROCEDURE inoutProc1( p1 INTEGER, p2 IN OUT INTEGER, p3
OUT INTEGER)
AS
BEGIN
    p2 := p1 + p2;
    p3 := p1 + 100;
END;
/
Create success.

```

아래는 inoutProc1을 실행하는 예를 보여준다.

```
iSQL> VAR v1 INTEGER
iSQL> VAR v2 INTEGER
iSQL> VAR v3 INTEGER
iSQL> EXEC :v1 := 3;
Execute success.
iSQL> EXEC :v2 := 5;
Execute success.
iSQL> EXEC inoutProc1(:v1, :v2, :v3);
Execute success.
iSQL> PRINT VAR;
[ HOST VARIABLE ]
```

NAME	TYPE	VALUE
..		
V1	INTEGER	3
V2	INTEGER	8
V3	INTEGER	103
..		

## 프로시저 삭제

프로시저를 삭제하는 기능을 제공한다.

다음은 emp\_proc를 삭제하는 예를 보여준다.

```
iSQL> DROP PROCEDURE emp_proc;
Drop success
```

## 함수 생성과 실행 및 삭제

### 함수 생성

함수를 생성하는 기능을 제공한다. 함수 생성시 반드시 아래 구문으로 끝나야 하며 리턴 타입이 정의되어 있어야 한다.

```
END;
/
```

생성된 함수는 sys\_procedures\_ 메타 테이블을 참조하여 확인할 수 있다.

다음은 UPDATE 문과 SELECT 문을 수행하는 함수 emp\_func를 생성하는 예를 보여준다.

```
iSQL> CREATE OR REPLACE FUNCTION emp_func(f1 IN INTEGER)
RETURN NUMBER
AS
  f2 NUMBER;
BEGIN
```

```

UPDATE employees SET salary = 1000000 WHERE eno = f1;
SELECT salary INTO f2 FROM employees WHERE eno = f1;
RETURN f2;
END;
/
Create success.

```

```

iSQL> SELECT * FROM system_.sys_procedures_;

```

```

USER_ID      PROC_OID      PROC_NAME

```

```

-----
OBJECT_TYPE STATUS      PARA_NUM      RETURN_DATA_TYPE RETURN_LANG_ID
-----
RETURN_SIZE RETURN_PRECISION RETURN_SCALE PARSE_NO      PARSE_LEN
-----
CREATED      LAST_DDL_TIME
-----

```

```

.
.
.
2          3300024          INOUTPROC1
0          0          3          2          132
15-SEP-2010 15-SEP-2010
2          3302344          EMP_FUNC
1          0          1          6          30000
23         38          0          3          209
15-SEP-2010 15-SEP-2010
36 rows selected.

```

## 함수 실행

함수를 실행하는 기능을 제공한다. 함수를 실행함으로써 다양한 쿼리를 한꺼번에 수행할 수 있다. 실행할 함수에 파라미터가 있는 경우 반드시 함수 실행전에 파라미터 개수만큼 변수가 선언되어 있어야 한다. 또한, 함수의 실행 결과를 저장할 변수도 정의되어 있어야 한다.

다음은 emp\_func를 실행하는 예를 보여준다.

```

iSQL> VAR eno INTEGER
iSQL> VAR ret NUMBER
iSQL> EXEC :eno := 11;
Execute success.
iSQL> EXEC :ret := emp_func(:eno);
Execute success.
iSQL> SELECT eno, salary FROM employees WHERE eno = 11;
ENO      SALARY
-----
11      1000000
1 row selected.

```



## 함수 삭제

함수를 삭제하는 기능을 제공한다.

다음은 emp\_func를 삭제하는 예를 보여준다.

```
iSQL> DROP FUNCTION emp_func;
Drop success
```

## 사용자 편의 기능

### 히스토리

이전에 수행했던 명령들의 리스트를 보여 준다.

해당 번호를 사용하여 이전에 수행했던 명령을 간단하게 실행할 수 있다.

```
iSQL> HISTORY; -> history 목록보기
```

또는

```
iSQL> H;
1 : SELECT * FROM tab;
2 : SELECT * FROM book;
3 : HISTORY;

iSQL> / -> 가장 마지막 명령(HISTORY;)을 재수행
iSQL> 2/      -> history 목록의 2번에 해당하는 명령(SELECT * FROM book;)실행
```

### 히스토리 저장

iSQL에서 실행한 명령어들을 iSQL 종료 시 파일로 자동 저장하는 기능이다. 이 기능을 활성화시키면 iSQL 재실행 시 파일에 저장된 이전 명령어들이 자동으로 로딩되기 때문에, 사용자가 화살표 키를 눌러서 이전 명령어에 접근 및 실행할 수 있다.

히스토리 저장 기능을 사용하려면 ISQL\_HIST\_FILE 환경변수를 설정하고 iSQL을 실행해야 한다.

```
$ export ISQL_HIST_FILE=~/.isql_history
```

히스토리 저장 기능을 끄기 위해서는 ISQL\_HIST\_FILE 환경변수를 삭제한다.

```
$ unset ISQL_HIST_FILE
```

### 기본값

사용 안 함

## 제약 사항

- 명령 프롬프트나 셸 프롬프트에서 키보드 방향키를 이용해 이전 명령어를 확인할 수 있는 경우에 사용 가능
- 최대 100개 저장 가능

사용자가 iSQL 프롬프트에서 입력한 모든 명령어가 파일에 저장되기 때문에 데이터베이스 사용자 암호 같은 민감한 정보도 유출될 수 있으므로 파일 접근 관리에 유의해야 한다.

## 셸 명령

iSQL에서 !다음에 바로 셸 명령을 수행할 수 있는 편리한 기능이다.

```
iSQL> !ls -al
total 3417
-rw-r----- 1 wlgml337 section 1198 Nov  1 13:30 .aliases
-rw----- 1 wlgml337 section 5353 Oct 18 21:17 .bash_history
-rw-r----- 1 wlgml337 section 1436 Nov  2 15:42 .bashrc
-rw-r----- 1 wlgml337 section 1549 Dec 13 17:36 .profile
drwxr-x--- 2 wlgml337 section 512 Nov  2 02:00 TEMP
drwxr-xr-x 2 root root 512 Oct 16 11:29 TT_DB
-rw----- 1 wlgml337 section 3446548 Dec 18 13:19 core
drwxr-x--- 2 wlgml337 section 512 Nov 11 16:33 cron
drwxr-x--- 2 wlgml337 section 512 Nov 15 10:52 test
drwxr-xr-x 6 wlgml337 section 512 Nov 11 11:45 work
```

## 명령 프롬프트

기본 명령 프롬프트인 'iSQL>' 대신 다른 값을 설정하여 프롬프트를 변경할 수 있다. 현재 접속한 사용자, 현재 시간 등의 런타임 변수가 포함된 경우 SET SQLPROMPT는 동적으로 변수를 치환한다.

```
SET SQLP[ROMPT] {text}
```

사용 가능한 치환 변수는 아래와 같다.

변수	설명
_CONNECT_IDENTIFIER	접속한 서버. "host:port_no"로 표현된다.
_DATE	현재 시간. DATE_FORMAT에 설정된 형식으로 표현된다.
_PRIVILEGE	iSQL 접속 권한을 보여준다. sysdba로 접속한 경우 '(sysdba)'로 치환된다.
_USER	현재 접속한 사용자 이름.

## 예제

```
iSQL>SET SQLPROMPT "_CONNECT_IDENTIFIER> "

iSQL>SET SQLP "_USER> "

iSQL>SET SQLPROMPT "_USER'@'_CONNECT_IDENTIFIER > "

iSQL>SET SQLPROMPT "_USER on _DATE from _CONNECT_IDENTIFIER> "
```

## 도움말

iSQL이 제공하는 명령에 대한 도움말을 제공한다. HELP 명령은 도움말 사용법을 보여 주며 특정 명령에 대한 도움말은 HELP 명령 다음에 정보를 알고자 하는 명령을 입력하면 된다.

```
iSQL> HELP;
Use 'help [command]'
Enter 'help index' for a list of command
iSQL> HELP INDEX;
/          EXIT          PARTITIONS
@          EXPLAINPLAN   QUERYLOGGING
ALTER      FEEDBACK      QUIT
AUTOCOMMIT FOREIGNKEYS   ROLLBACK
CHKCONSTRAINTS FULLNAME  SAVE
CL[EAR]    H[ISTORY]     SELECT
COL[UMN]   HEADING       SPOOL
COLSIZE    INSERT        SQLP[ROMPT]
COMMIT     LINESIZE      START
CREATE     LOAD          TERM
DEFINE     LOBOFFSET     TIMESCALE
DELETE     LOBSIZE       TIMING
DESC       MERGE         UPDATE
DROP       MOVE          USER
ECHO       NUM[WIDTH]    VAR[TABLE]
EDIT       NUMF[ORMAT]   VERTICAL
EXECUTE    PAGESIZE
```

```
iSQL> HELP EXIT;
exit;
or
quit; - exit iSQL
```

## 내셔널 캐릭터 사용법

NCHAR 및 NVARCHAR 타입의 내셔널 캐릭터 상수 문자를 사용하기 위해서 아래와 같은 방법으로 환경변수 등을 설정해야 데이터의 손실 염려가 없다.

- 환경변수 ALTIBASE\_NLS\_NCHAR\_LITERAL\_REPLACE의 값을 1로 설정한다.

```
$ export ALTIBASE_NLS_NCHAR_LITERAL_REPLACE =1
```

- SQL 구문에서 NCHAR 타입 상수 문자열을 사용하기 위해 해당 문자열 바로 앞에 "N"을 붙여 사용한다.

```
isQL> create table t1 (c1 nvarchar(10));  
Create success.  
isQL> insert into t1 values (N'AB가나');  
1 row inserted.  
isQL> select * from t1;  
c1  
-----  
AB가나  
1 row selected.
```