참조 설명서
제 1권: Building Blocks

Sybase® Adaptive Server® Enterprise

12.5

문서 번호: 32155-01-1250-01
최종 개정 일자: 2001년 6월

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

목차

설명서 정보

Adaptive Server 참조 설명서는 Sybase® Adaptive Server® Enterprise와 Transact-SQL® 언어에 대한 총 네 권의 설명서로 구성됩니다.

제 1권 "*Building Blocks*"는 데이터 유형, 내장 함수, 표현식과 식별자, 예약어, SQLSTATE 에러 등 Transact-SQL의 "일부"를 설명합니다. Transact-SQL을 제대로 사용하려면 Building Blocks의 작업 내용과 이러한 작업이 Transact-SQL 구문의 결과에 미치는 영향을 이해해야 합니다.

제 2권 "*명령*"은 구문 생성에 사용하는 Transact-SQL 명령의 참조 정보를 제공합니다.

제 3권 "*프로시저*"는 시스템 프로시저, 카탈로그 내장 프로시저, 확장 내장 프로시저와 dbcc 내장 프로시저에 대한 참조 정보를 제공합니다. 모든 프로시저는 Transact-SQL 구문을 사용하여 생성됩니다.

제 4권 "*시스템 테이블*"은 서버, 데이터베이스, 사용자 및 기타 정보를 저장하는 시스템 테이블에 대한 참조 정보를 제공하며 dbccdb와 dbccalt 데이터베이스의 테이블에 대한 정보를 제공합니다.

본 설명서의 사용자

이 *Adaptive Server 참조 설명서*는 모든 수준의 Transact-SQL 사용자를 대상으로 합니다.

설명서 이용 방법

- 1장 "시스템 및 사용자 정의 데이터 유형"에서는 Adaptive Server와 함께 제공되는 사용자 정의 데이터 유형과 시스템에 대해 설명하며 사용자 정의 데이터 유형을 생성하는 방법을 설명합니다.
- 2장 "Transact-SQL 함수"에서는 이름과 간략한 설명이 있는 표를 통해 Adaptive Server 함수를 나열합니다. 해당 함수로 가려면 표에서 함수 이름을 누릅니다.
- 3장부터 6장까지는 개별적인 함수에 대한 설명서 페이지입니다.
- 7장 "표현식, 식별자 및 대표 문자"에서는 Transact-SQL 언어 사용에 대한 정보를 제공합니다.

- 8장 "예약어"에서는 Transact-SQL 및 SQL92 키워드에 대한 정보를 제공합니다.
- 9장 "SQLSTATE 코드 및 메시지"에는 Adaptive Server의 SQLSTATE 상태 코드 및 관련 메시지에 대한 정보가 들어 있습니다.

관련 문서

Sybase Adaptive Server Enterprise 설명서는 다음과 같이 구성되어 있습니다.

- 해당 플랫폼용 릴리스 정보 – 이 설명서에 포함되지 않은 최신 정보가 들어 있습니다.

  최신 릴리스 정보는 웹을 통해 이용할 수 있습니다. 이 제품 CD 발매 후에 추가된 중요한 제품이나 자료에 대한 정보를 보려면 Sybase Technical Library를 이용하십시오.
- 해당 플랫폼용 *설치 안내서* – 모든 Adaptive Server와 관련 Sybase 제품의 설치, 업그레이드, 구성 절차에 대해 설명합니다.
- 해당 플랫폼용 *Adaptive Server Enterprise 구성 안내서* – Adaptive Server의 특정 구성 작업을 수행하는 지침에 대해 설명합니다.
- *Adaptive Server Enterprise 새 기능 설명서* – Adaptive Server 버전 12.5의 새로운 기능과 이러한 기능을 지원하기 위해 추가된 시스템 변경 사항 및 기존 응용 프로그램에 미치는 영향에 대해 설명합니다.
- *Transact-SQL User's Guide* – 관계형 데이터베이스 언어의 Sybase 확장 버전인 Transact-SQL에 대해 설명합니다. 이 설명서는 데이터베이스 관리 시스템을 처음 사용하는 사용자에게 교과서 역할을 합니다. 이 설명서에는 pubs2 및 pubs3 예제 데이터베이스에 대한 설명도 포함되어 있습니다.
- *시스템 관리 지침서* – 서버와 데이터베이스 관리 방법에 대해 자세히 설명합니다. 이 설명서에는 물리적 자원, 보안, 사용자 및 시스템 데이터베이스를 관리하는 방법과 문자 변환, 다국어, 정렬 순서를 지정하는 방법이 포함되어 있습니다.
- *참조 설명서* – 모든 Transact-SQL 명령, 함수, 프로시저, 데이터 유형에 대해 자세히 설명합니다. 또한 Transact-SQL 예약어 목록과 시스템 테이블의 정의를 제공합니다.
- *Performance and Turning Guide* – 최대의 성능을 사용할 수 있도록 Adaptive Server를 튜닝하는 방법을 소개합니다. 이 설명서는 성능에 영향을 주는 데이터베이스의 설계 문제, 쿼리 최적화, 대용량 데이터베이스에서의 Adaptive Server 튜닝 방법, 디스크와 캐시 문제, lock과 커서가 성능에 주는 영향에 대해 설명합니다.

- *유틸리티 안내서* – 운영 체제 수준에서 실행되는 isql과 bcp 등의 Adaptive Server 유틸리티 프로그램에 대해 설명합니다.
- *요약 참조 설명서* – 명령, 함수, 시스템 프로시저, 확장 시스템 프로시저, 데이터 유형, 유틸리티의 이름 및 구문 목록을 포켓 크기의 책으로 제공합니다. 인쇄 버전만 있습니다.
- *시스템 테이블 다이어그램* – 시스템 테이블 및 항목 관계를 포스터 형식으로 설명합니다. 인쇄 버전만 있습니다.
- *Error Messages and Troubleshooting Guide* – 자주 발생하는 에러 메시지와 시스템 문제에 대한 해결 방법을 설명합니다.
- *Component Integration Services User's Guide* – Adaptive Server의 CIS(Component Integration Services) 기능을 사용하여 원격 Sybase 데이터베이스 및 Sybase 이외의 데이터베이스에 연결하는 방법을 설명합니다.
- *Java in Adaptive Server Enterprise* – Adaptive Server 데이터베이스에서 데이터 유형과 사용자 정의 함수로 Java 클래스를 설치하고 사용하는 방법에 대해 설명합니다.
- *Using Sybase Failover in a High Availability System* – 고가용성(HA) 시스템에서 Sybase Failover를 사용하여 Adaptive Server를 companion 서버로 구성하는 방법에 대해 설명합니다.
- *Using Adaptive Server Distributed Transaction Management Features* – 분산 트랜잭션 처리 환경에서 Adaptive Server DTM 기능을 구성 및 사용하고 문제를 해결하는 방법에 대해 설명합니다.
- *EJB Server User's Guide* – EJB Server를 사용하여 Adaptive Server에서 Enterprise JavaBeans를 배치하고 실행하는 방법에 대해 설명합니다.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – X/Open XA 트랜잭션 관리자와 함께 Sybase DTM XA 인터페이스를 사용하는 방법에 대해 설명합니다.
- *용어집* – Adaptive Server 설명서에서 사용된 기술 용어를 정의합니다.
- *Sybase jConnect for JDBC Programmer's Reference* – jConnect for JDBC 제품과 이를 사용하여 관계형 데이터베이스 관리 시스템에 저장된 데이터에 액세스하는 방법을 설명합니다.
- *Full-Text Search Specialty Data Store User's Guide* – Adaptive Server Enterprise 데이터를 검색하기 위하여 Verity를 갖는 전체 텍스트 검색 기능을 사용하는 방법에 대해 설명합니다.

- *Historical Server User's Guide* − Historical Server를 사용하여 SQL Server와 Adaptive Server의 성능 정보를 얻는 방법에 대해 설명합니다.
- *Monitor Server User's Guide* − Monitor Server를 사용하여 SQL Server와 Adaptive Server에서 성능 통계를 얻는 방법에 대해 설명합니다.
- *Monitor Client Library Programmer's Guide* − Adaptive Server 성능 데이터에 액세스하는 Monitor Client Library 응용 프로그램을 작성하는 방법에 대해 설명합니다.

추가 정보

제품에 대한 자세한 내용은 다음의 Sybase Technical Library CD와 Technical Library Product Manuals 웹 사이트를 참조하십시오.

- Technical Library CD에는 제품 설명서와 소프트웨어가 함께 들어 있습니다. http://www.sybase.com/detail/1,3693,1010661,00.html의 Product Manuals에서 다운로드할 수 있는 DynaText 브라우저를 사용하면 제품에 관한 기술 정보를 손쉽게 찾아볼 수 있습니다.

  Technical Library를 설치하고 시작하는 방법은 설명서 패키지에서 *Technical Library Installation Guide*를 참고하십시오.

- Technical Library Product Manuals 웹 사이트는 Technical Library CD의 HTML 버전으로서 일반 웹 브라우저를 사용하여 액세스할 수 있습니다. 제품 설명서 외에도 Technical Documents 웹 사이트(이전에는 Tech Info Library)와 Solved Cases 페이지, Sybase/Powersoft 뉴스그룹에 대한 링크가 포함되어 있습니다.

  Technical Library Product Manuals 웹 사이트를 방문하려면 http://www.sybase.com/support/manuals/의 Product Manuals로 이동하십시오.

본 설명서의 표기법

다음은 본 설명서에 사용된 표기법에 대한 설명입니다.

SQL은 자유로운 형식의 언어입니다. 한 줄에 들어갈 수 있는 단어의 수 또는 줄을 끝내야 하는 위치에 대해 특별히 지정된 규칙은 없습니다. 그러나 읽기 쉽게 하기 위해서 이 설명서의 모든 예제와 구문은 구문의 각 절이 새로운 줄에서 시작하도록 설정되어 있습니다. 여러 부분으로 구성된 절은 다음 줄로 넘기면서 들여쓰기가 되어 있습니다. 복잡한 명령은 수정 BNF(Backus Naur Form) 표기법을 사용하여 설정되어 있습니다.

표 1은 본 설명서에 있는 구문 문장의 표기법입니다.

표 1: 글꼴 및 구문 표기법

요소	예제
명령 이름, 명령 옵션, 유틸리티 이름, 유틸리티 옵션 그리고 기타 키워드는 굵게 표시됩니다.	select sp_configure
데이터베이스 이름, 데이터 유형, 파일 이름, 경로 이름은 이탤릭체로 표시됩니다.	master 데이터베이스

변수 등 사용자가 입력해야 하는 값을 나타내는 단어는 이탤릭체로 표시됩니다.	`select column_name`
	`from table_name`
	`where search_conditions`

명령의 한 부분은 괄호로 입력합니다.	`compute row_aggregate (column_name)`
더블 콜론 및 등호는 구문이 BNF 표기법으로 되어 있음을 나타냅니다. 실제로 옵션을 입력할 때에는 더블 콜론이나 등호를 사용하지 않습니다. "다음과 같이 정의됨"을 나타냅니다.	::=
중괄호 안에 포함된 옵션은 최소한 하나를 선택해야 합니다. 실제로 옵션을 입력할 때에는 중괄호를 사용하지 않습니다.	`{cash, check, credit}`
대괄호 안에 포함된 옵션은 하나 이상을 선택하는 선택 사항입니다. 실제로 옵션을 입력할 때에는 대괄호를 입력하지 않습니다.	`[cash | check | credit]`
콤마는 표시된 옵션을 원하는 만큼 선택할 수 있다는 것을 의미합니다. 선택한 옵션 사이에 콤마를 넣으십시오.	`cash, check, credit`
파이프 또는 세로줄(|)은 표시된 옵션 중 하나만 선택할 수 있다는 것을 의미합니다.	`cash | check | credit`

생략 기호(...)는 마지막 단위를 원하는 만큼 *반복*할 수 있다는 것을 의미합니다.	`buy thing = price [cash | check | credit]`
	`[, thing = price [cash | check | credit]]...`
	최소한 한 개를 구입해야 하며 가격을 지불해야 합니다. 대괄호 내의 항목 중 지불 방법을 선택할 수 있습니다. 추가로 다른 항목을 원하는 만큼 구입할 수 있습니다. 각 구입 품목마다 이름, 가격, 지불 방법(선택 사항)을 입력합니다.

- 명령의 구문과 모든 옵션을 나타내는 구문은 다음과 같습니다

  sp_dropdevice [*device_name*]

  또는 여러 옵션을 사용하는 명령의 경우는 다음과 같습니다.

  select *column_name*
  from *table_name*
  where *search_conditions*

구문 문장에서 키워드(명령어)는 보통 글꼴이며 식별자는 소문자입니다. 이탤릭체는 사용자 입력어를 나타냅니다.

- Transact-SQL 명령의 사용 예제는 다음과 같이 출력됩니다.

```
select * from publishers
```

- 컴퓨터에서 출력된 항목의 예는 다음과 같습니다.

```
pub_id    pub_name                city          state
-------   ---------------------   -----------   -----
0736      New Age Books           Boston        MA
0877      Binnet & Hardley        Washington    DC
1389      Algodata Infosystems    Berkeley      CA

(3 rows affected)
```

이 설명서에서 예제의 대부분은 소문자로 되어 있습니다. 그러나 Transact-SQL 키워드를 입력할 때는 대/소문자를 무시할 수 있습니다. 예를 들어, SELECT, Select, select는 모두 동일합니다.

Adaptive Server에서 테이블 이름 등 데이터베이스 객체에 대한 대/소문자 구분 여부는 Adaptive Server에 설치된 정렬 순서에 따라 다릅니다. 단일 바이트 문자 집합에서 대/소문자 구분을 변경하여 Adaptive Server의 정렬 순서를 재구성할 수 있습니다. 자세한 내용은 *시스템 관리 지침서*를 참조하십시오.

지원 요청

Sybase 제품 지원 계약을 맺으면 Sybase 기술 지원부 직원 중 1인이 전담 지원 담당으로 선정됩니다. 온라인 도움말이나 설명서를 사용해도 문제를 해결할 수 없는 경우에는 Sybase 기술 지원부 또는 가까운 Sybase 지점에 연락하십시오.

1 장

시스템 및 사용자 정의 데이터 유형

이 장에서는 Transact-SQL 데이터 유형에 대해 설명합니다. 데이터 유형은 열, 내장 프로시저 매개변수, 로컬 변수 등의 유형, 크기 및 저장 형식을 지정합니다. 이 장에서 다루는 항목은 다음과 같습니다.

데이터 유형 범주

Adaptive Server는 여러 가지 시스템 데이터 유형과 사용자 정의 데이터 유형인 timestamp 및 sysname을 제공합니다. 표 1-1은 Adaptive Server 데이터 유형의 범주를 보여 줍니다. 각 범주에 대해서는 이 장의 해당 절에서 설명합니다.

표 1-1: 데이터 유형 범주

범주	용도
정밀 숫자 데이터 유형	정밀하게 표현해야 하는 숫자 값(정수 및 소수부를 가진 숫자 모두)
근사 숫자 데이터 유형	산술 연산 시 반올림을 허용하는 숫자 데이터
통화 데이터 유형	통화 데이터
타임스탬프 데이터 유형	Client-Library™ 응용 프로그램에서 찾는 테이블
날짜 및 시간 데이터 유형	날짜 및 시간 정보
문자 데이터 유형	문자, 숫자 및 기호로 구성된 문자열
Binary 데이터 유형	그림과 같이 16진수와 같은 표기법 형식의 raw 이진 데이터
bit 데이터 유형	True/false 및 yes/no 유형 데이터
sysname 데이터 유형	시스템 테이블
text 및 image 데이터 유형	서버의 논리적 페이지 크기에 따라 제공되는 최대 열 크기보다 더 필요한 인쇄 가능한 문자 또는 16진수와 같은 데이터
사용자 정의 데이터 유형	규칙, 디폴트, NULL 유형, IDENTITY 속성, 기본 데이터 유형을 상속하는 객체 정의

범위 및 저장 장소 크기

표 1-2는 시스템에서 제공하는 데이터 유형과 그 동의어를 보여 주며 각각에 대한 유효 값의 범위와 저장 장소 크기에 대한 정보를 제공합니다. Adaptive Server에서는 시스템 데이터 유형에 대문자나 소문자 중 한 가지를 사용할 수 있지만 데이터 유형을 인쇄하면 간단하게 표시하기 위해 소문자로 인쇄됩니다. timestamp와 같은 사용자 정의 데이터 유형은 *대/소문자를 구분*합니다. Adaptive Server에서 제공하는 대부분의 데이터 유형은 예약된 단어가 아니며 다른 객체의 이름에 사용할 수 있습니다.

표 1-2: 시스템 데이터 유형의 범위와 저장 장소 크기

데이터 유형	동의어	범위	저장 장소의 바이트
정밀 숫자 데이터 유형			

데이터 유형	동의어	범위	저장 장소의 바이트
tinyint		0 ~ 255	1
smallint		-2^{15} (-32,768) ~ 215 $^{-1}$ (32,767)	2
int	integer	-2^{31} (-2,147,483,648) ~ 2^{31} -1 (2,147,483,647)	4
numeric (p, s)		-10^{38} ~ 10^{38} -1	2 ~ 17
decimal (p, s)	dec	-10^{38} ~ 10^{38} -1	2 ~ 17
근사 숫자 데이터 유형			
float (precision)		시스템 종속적	4 또는 8
double precision		시스템 종속적	8
real		시스템 종속적	4
통화 데이터 유형			
smallmoney		-214,748.3648 ~ 214,748.3647	4
money		-922,337,203,685,477.5808 ~ 922,337,203,685,477.5807	8
날짜/시간 데이터 유형			
smalldatetime		1900년 1월 1일 ~ 2079년 6월 6일	4
datetime		1753년 1월 1일 ~ 9999년 12월 31일	8
문자 데이터 유형			
char(n)	character	서버의 논리적 페이지 크기에 대한 최대 열 크기에 의해 정해짐	n
varchar(n)	char[acter] varying	서버의 논리적 페이지 크기에 대한 최대 열 크기에 의해 정해짐	실제 엔트리 길이
unichar	Unicode character	서버의 논리적 페이지 크기에 대한 최대 열 크기에 의해 정해짐	n*@@unicharsize (@@unicharsize equals 2)
univarchar	unichar(acter) varying	서버의 논리적 페이지 크기에 대한 최대 열 크기에 의해 정해짐	문자의 실제 수 *@@unicharsize
nchar(n)	national char[acter]	서버의 논리적 페이지 크기에 대한 최대 열 크기에 의해 정해짐	n * @@ncharsize
nvarchar(n)	nchar varying, national char[acter] varying	서버의 논리적 페이지 크기에 대한 최대 열 크기에 의해 정해짐	n
Binary 데이터 유형			
binary(n)		서버의 논리적 페이지 크기에 대한 최대 열 크기에 의해 정해짐	n
varbinary(n)		서버의 논리적 페이지 크기에 대한 최대 열 크기에 의해 정해짐	실제 엔트리 길이
비트 데이터 유형			

데이터 유형	동의어	범위	저장 장소의 바이트
bit		0 또는 1	1(1바이트는 최대 8bit 열 포함)
텍스트 및 이미지 데이터 유형			
text		2^{31} -1 (2,147,483,647)바이트 이하	초기화될 때까지는 0, 그 다음에는 논리적 페이지 크기의 배수
image		2^{31} -1 (2,147,483,647)바이트 이하	초기화될 때까지는 0, 그 다음에는 논리적 페이지 크기의 배수

열, 변수 또는 매개변수의 데이터 유형 선언

열, 로컬 변수 또는 매개변수의 데이터 유형을 선언해야 합니다. 데이터베이스의 데이터 유형은 시스템에서 제공하는 데이터 유형이거나 사용자 정의된 데이터 유형이 될 수 있습니다.

테이블의 열에 대한 데이터 유형 선언

다음 구문을 사용하여 create table 또는 alter table 구문에 있는 새 열의 데이터 유형을 선언하십시오.

```
create table [[database.]owner.]table_name
    (column_name datatype [identity | not null | null]
        [, column_name datatype [identity | not null |
            null]]...)

alter table [[database.]owner.]table_name
    add column_name datatype [identity | null
        [, column_name datatype [identity | null]...
```

예를 들어 다음과 같습니다.

```
create table sales_daily
    (stor_id char(4)not null,
     ord_num numeric(10,0)identity,
     ord_amt money null)
```

배치 또는 프로시저에의 로컬 변수에 대한 데이터 유형 선언

다음 구문을 사용하여 배치나 내장 프로시저에 있는 로컬 변수에 대한 데이터 유형을 선언하십시오.

```
declare @variable_name datatype
    [, @variable_name datatype]...
```

예를 들어 다음과 같습니다.

```
declare @hope money
```

내장 프로시저의 매개변수에 대한 데이터 유형 선언

다음 구문을 사용하여 내장 프로시저의 매개변수에 대한 데이터 유형을 선언하십시오.

```
create procedure [owner.]procedure_name [;number]
    [[()@parameter_name datatype [= default] [output]
        [,@parameter_name datatype [= default]
            [output]]...[)]]
[with recompile]
as SQL_statements
```

예를 들어 다음과 같습니다.

```
create procedure auname_sp @auname varchar(40)
as
    select au_lname, title, au_ord
    from authors, titles, titleauthor
    where @auname = au_lname
    and authors.au_id = titleauthor.au_id
    and titles.title_id = titleauthor.title_id
```

리터럴의 데이터 유형 결정

사용자는 리터럴의 데이터 유형을 선언할 수 없습니다. Adaptive Server는 모든 문자 리터럴을 varchar로 취급합니다. E 표기법으로 입력한 숫자 리터럴은 float로 취급되며 그 밖의 숫자 리터럴은 모두 정밀 숫자로 취급됩니다.

- 소수점이 없는 2^{31} - 1과 -2^{31} 사이의 리터럴은 integer로 취급됩니다.

- 소수점을 포함하거나 정수의 범위 밖에 있는 리터럴은 numeric으로 취급됩니다.

참고 이전 버전과의 호환성을 유지하려면 float로 취급해야 하는 숫자 리터럴에 E 표기법을 사용하십시오.

혼합 모드 표현식의 데이터 유형

여러 가지 데이터 유형을 가진 값에서 연결이나 혼합 모드 산술을 수행하는 경우 Adaptive Server는 결과의 데이터 유형, 길이, 정밀도(P)를 결정해야 합니다.

데이터 유형 계층 결정

각 시스템 데이터 유형은 systypes 시스템 테이블에 저장되는 *데이터 유형 계층*을 갖습니다. 사용자 정의 데이터 유형은 기준이 되는 시스템 데이터 유형의 계층을 상속합니다.

다음 쿼리는 데이터베이스의 데이터 유형을 계층에 따라 분류합니다. 아래 표시된 정보 외에 쿼리 결과에는 데이터베이스에 있는 사용자 정의 데이터 유형에 대한 정보가 포함됩니다.

```
select name,hierarchy
from systypes
order by hierarchy
name                          hierarchy
----------------------------  ---------
floatn                                1
float                                 2
datetimn                              3
datetime                              4
real                                  5
numericn                              6
numeric                               7
decimaln                              8
decimal                               9
moneyn                               10
money                                11
smallmoney                           12
smalldatetime                        13
```

```
intn                                    14
int                                     15
smallint                                16
tinyint                                 17
bit                                     18
univarchar                              19
unichar                                 20
reserved                                21
varchar                                 22
sysname                                 22
nvarchar                                22
char                                    23
nchar                                   23
varbinary                               24
timestamp                               24
binary                                  25
text                                    26
image                                   27
(28 rows affected)
```

데이터 유형 계층은 여러 데이터 유형 값을 사용하여 계산의 결과를 결정하며 결과 값은 목록의 맨 위에 가장 가까운 데이터 유형에 할당됩니다.

다음 예제에서 sales 테이블의 *qty*는 roysched의 royalty를 곱한 값입니다. qty는 smallint이고 16의 계층을 가지며 royalty는 int이고 15의 계층을 갖습니다. 따라서 결과의 데이터 유형은 int입니다.

smallint(qty) * int(royalty) = int

정밀도 (P) 및 소수점 이하 자릿수 결정

numeric 및 decimal 데이터 유형의 경우 정밀도(P)와 소수점 이하 자릿수의 각 조합은 별개의 Adaptive Server 데이터 유형입니다. 두 개의 numeric 또는 decimal 값에서 산술을 수행할 경우

- *n1*은 정밀도(P) *p1*과 소수점 이하 자릿수 *s1*을 가지며
- *n2*는 정밀도(P) *p2*와 소수점 이하 자릿수 *n2*를 갖습니다.

Adaptive Server는 결과의 정밀도(P)와 소수점 이하 자릿수를 표 1-3에서와 같이 결정합니다.

표 1-3: 산술 연산 후의 정밀도(P) 및 소수점 이하 자릿수

연산	정밀도(P)	소수점 이하 자릿수
n1 + n2	max(s1, s2) + max(p1 -s1, p2 - s2) + 1	max(s1, s2)
n1 - n2	max(s1, s2) + max(p1 -s1, p2 - s2) + 1	max(s1, s2)
n1 * n2	s1 + s2 + (p1 - s1) + (p2 - s2) + 1	s1 + s2
n1 / n2	max(s1 + p2 + 1, 6) + p1 - s1 + p2	max(s1 + p2 -s2 + 1, 6)

하나의 데이터 유형을 다른 데이터 유형으로 변환

하나의 데이터 유형에서 다른 데이터 유형으로의 변환 중 대부분은 Adaptive Server가 자동으로 처리하는데 이를 implicit conversion이라고 합니다. 기타 변환은 convert, inttohex 및 hextoint 함수를 사용하여 명시적으로 수행해야 합니다. Adaptive Server에서 지원하는 데이터 유형 변환에 대한 자세한 내용은 "데이터 유형 변환 함수"를 참조하십시오.

고정 길이 NULL 열의 자동 변환

가변 길이 데이터 유형을 가진 열만 null 값을 저장할 수 있습니다. 고정 길이 데이터 유형을 가진 NULL 열을 생성하면 Adaptive Server는 이를 대응하는 가변 길이 데이터 유형으로 자동 변환합니다. Adaptive Server는 사용자에게 데이터 유형이 변경되었음을 알리지 않습니다.

표 1-4는 변환할 고정 길이 및 가변 길이 데이터 유형을 보여 줍니다. moneyn과 같은 일부 가변 길이 데이터 유형은 예약된 데이터 유형으로 열, 변수, 매개변수를 생성하는 데 사용할 수 없습니다.

표 1-4: 고정 길이 데이터 유형의 자동 변환

원래 고정 길이 데이터 유형	변환 대상
char	varchar
unichar	univarchar
nchar	nvarchar
binary	varbinary
datetime	datetimn
float	floatn
int, smallint 및 tinyint	intn
decimal	decimaln
numeric	numericn
money 및 smallmoney	moneyn

오버플로 및 truncation 에러 처리

arithabort 옵션은 산술 에러 발생 시 Adaptive Server의 동작을 결정합니다. 두 개의 arithabort 옵션인 arithabort arith_overflow와 arithabort numeric_truncation은 다른 유형의 산술 에러를 처리합니다. 사용자는 각 옵션을 개별적으로 설정하거나 단일 set arithabort on 또는 set arithabort off 문을 사용하여 두 옵션을 설정할 수 있습니다.

- arithabort arith_overflow는 explicit 또는 implicit datatype conversion 동안 0으로 나누기 에러 또는 정밀도(P) 손실이 발생할 경우의 동작을 지정합니다. 이 에러 유형은 심각한 것으로 간주됩니다. 디폴트 설정인 arithabort arith_overflow on은 에러가 발생하는 전체 트랜잭션을 롤백(roll back)합니다. 트랜잭션을 포함하지 않는 배치에서 에러가 발생할 경우 arithabort arith_overflow on은 배치에서 초기 명령을 롤백(roll back)하지만 Adaptive Server는 배치에서 에러 생성 구문 뒤에 오는 구문을 실행하지 않습니다.

  arithabort arith_overflow off를 설정할 경우 Adaptive Server는 에러가 발생한 문을 중지하지만 트랜잭션이나 배치의 다른 구문을 계속해서 처리합니다.

- arithabort numeric_truncation은 implicit datatype conversion 동안 정밀 숫자 데이터 유형에 의한 소수점 이하 자릿수 손실이 발생할 경우의 동작을 지정합니다. explicit conversion에서 소수점 이하 자릿수가 손실되는 경우 경고 없이 결과는 truncation됩니다. 디폴트 설정인 arithabort numeric_truncation on은 에러를 일으키는 구문은 중단하지만 트랜잭션이나 배치의 다른 구문은 계속 처리합니다. arithabort numeric_truncation off를 설정하면 Adaptive Server는 쿼리 결과를 truncation하고 처리를 계속합니다.

arithignore 옵션은 Adaptive Server에 오버플로 에러가 발생한 후에 경고 메시지를 출력할 것인지 여부를 결정합니다. 기본적으로 arithignore 옵션은 off 상태입니다. 이 상태에서는 Adaptive Server가 숫자 오버플로를 발생시키는 쿼리가 있을 경우 경고 메시지를 표시합니다. 오버플로 에러를 무시하려면 set arithignore on을 사용하십시오.

참고 arithabort 및 arithignore 옵션은 릴리스 10.0에 대해 재정의되었습니다. 응용 프로그램에서 이 옵션을 사용하는 경우 원하는 결과가 제대로 나타나는지 확인하기 위해 이 옵션을 검사합니다.

표준 및 호환성

표준	호환성 수준
SQL92	Transact-SQL은 *smallint*, *int*, *numeric*, *decimal*, *float*, *double precision*, *real*, *char* 및 *varchar* SQL92 데이터 유형을 제공합니다. *tinyint*, *binary*, *varbinary*, *image*, *bit*, *datetime*, *smalldatetime*, *money*, *smallmoney*, *nchar*, *nvarchar*,*unichar*, *univarchar*, *sysname*, *text*, *timestamp* 및 사용자 정의 데이터 유형은 Transact-SQL 확장입니다.

정밀 숫자 데이터 유형

기능

값을 정확히 표현하는 것이 중요한 경우 정밀 숫자 데이터 유형을 사용하십시오. Adaptive Server는 정수(전체 숫자) 및 소수점 이하 부분이 있는 숫자 모두에 대해 정밀 숫자 유형을 제공합니다.

정수 유형

Adaptive Server는 정수를 저장하는 세 가지 정밀 숫자 유형인 int(또는 integer), smallint 및 tinyint를 제공합니다. 저장할 숫자의 예상 크기에 따라 정수 유형을 선택하십시오. 내부 저장 장소 크기는 표 1-5에서와 같이 유형에 따라 다릅니다.

표 1-5: 정수 데이터 유형

데이터 유형	저장 장소	저장 장소 바이트
int[eger]	-2^{31}과 $2^{31} - 1$ (-2,147,483,648과 2,147,483,647) 사이의 정수, 포함	4
smallint	-2^{15}와 $2^{15} - 1$ (-32,768과 32,767) 사이의 정수, 포함	2
tinyint	0과 255 사이의 정수, 포함(음수는 허용되지 않습니다.)	1

정수 데이터 입력

정수 데이터를 콤마 없이 숫자 문자열로 입력합니다. 정수 데이터는 소수점 오른쪽의 모든 자릿수가 0인 경우 소수점을 포함할 수 있습니다. smallint 및 integer 데이터 유형에는 옵션으로 더하기 기호 또는 빼기 기호가 앞에 올 수 있습니다. tinyint 데이터 유형에는 옵션으로 더하기 기호가 앞에 올 수 있습니다.

표 1-6은 데이터 유형이 integer인 열에 유효한 몇 가지 엔트리를 보여주며 isql이 이 값을 어떻게 표시하는지 나타냅니다.

표 1-6: 유효한 정수 값

입력한 값	표시된 값
2	2
+2	2
-2	-2
2.	2
2.000	2

표 1-7은 integer 열에 유효하지 않은 몇 가지 엔트리를 보여 줍니다.

표 1-7: 유효하지 않은 정수 값

입력한 값	에러 유형
2,000	콤마는 허용되지 않습니다.
2-	빼기 기호는 숫자 앞에 와야 합니다.
3.45	소수점 오른쪽의 숫자는 0이 아닌 숫자입니다.

Decimal datatypes

Adaptive Server는 소수점이 포함된 숫자에 대해 두 가지 정밀 숫자 데이터 유형인 numeric 및 dec[imal]을 제공합니다. numeric 및 decimal 열에 저장된 데이터는 디스크 공간을 절약하기 위해 압축되며 산술 연산 후에 최소의 유효 자릿수까지 정밀도를 보존합니다. numeric 및 decimal 데이터 유형은 한 가지만 제외하고 모든 점에서 동일합니다. 소수점 이하 자릿수가 0인 numeric 데이터 유형만 IDENTITY 열에 사용할 수 있습니다.

정밀도(P) 및 소수점 이하 자릿수 지정

numeric 및 decimal 데이터 유형은 괄호로 둘러싸고 콤마로 구분되는 두 개의 옵션 매개변수인 precision과 scale을 사용합니다.

datatype [(*precision* [, *scale*])]

Adaptive Server는 정밀도(P)와 소수점 이하 자릿수의 조합을 별개의 데이터 유형으로 취급합니다. 예를 들어, numeric(10,0)과 numeric(5,0)은 별개의 두 데이터 유형입니다. precision과 scale은 decimal 또는 numeric 열에 저장할 수 있는 값의 범위를 결정합니다.

- 정밀도(P)는 열에 저장할 수 있는 소수 자릿수의 최대 수를 지정합니다. 소수점 오른쪽과 왼쪽에 있는 *모든* 자릿수가 포함되며, 1에서 38자릿수까지 범위 내에서 정밀도(P)를 지정하거나 디폴트 정밀도(P)인 18자릿수를 사용할 수 있습니다.

- 소수점 이하 자릿수는 소수점 오른쪽에 저장할 수 있는 자릿수의 최대 수를 지정합니다. 소수점 이하 자릿수는 정밀도(P)보다 작거나 같아야 합니다. 0에서 38자릿수 범위 내에서 소수점 이하 자릿수를 지정하거나 디폴트 소수점 이하 자릿수인 0자릿수를 사용할 수 있습니다.

저장 장소 크기

numeric 또는 decimal 열의 저장 장소 크기는 정밀도(P)에 따라 다릅니다. 최소 저장 장소 요구 사항은 1 또는 2자릿수 열에 대해 2바이트입니다. 저장 장소 크기는 추가 2자릿수 정밀도(P)에 대해 최대 17바이트까지 약 1바이트씩 증가합니다.

다음 수식을 사용하여 numeric 또는 decimal 열에 대한 정확한 저장 장소 크기를 계산할 수 있습니다.

```
ceiling (precision / log 256 ) + 1
```

예를 들어, numeric(18,4) 열에 대한 저장 장소 크기는 9바이트입니다.

Decimal 데이터 입력

decimal 및 numeric 데이터를 옵션 더하기 기호나 빼기 기호가 앞에 오고 옵션인 소수점을 포함하는 문자열로 입력합니다. 값이 열에 지정된 정밀도(P)나 소수점 이하 자릿수를 초과할 경우 Adaptive Server는 에러 메시지를 return합니다. 소수점 이하 자릿수가 0인 정밀 숫자 유형은 소수점 없이 표시됩니다.

표 1-8은 numeric(5,3)의 데이터 유형을 가진 열에 유효한 몇 가지 엔트리를 보여 주며 isql이 이 값을 어떻게 표시하는지 나타냅니다.

표 1-8: 유효한 decimal 값

입력한 값	표시된 값
12.345	12.345
+12.345	12.345
-12.345	-12.345
12.345000	12.345
12.1	12.100
12	12.000

표 1-9는 numeric(5,3)의 데이터 유형을 가진 열에 유효하지 않은 몇 가지 엔트리를 보여 줍니다.

표 1-9: 유효하지 않은 decimal 값

입력한 값	에러 유형
1,200	콤마는 허용되지 않습니다.
12-	빼기 기호는 숫자 앞에 와야 합니다.
12.345678	소수점 오른쪽에 너무 많은 0이 아닌 자릿수가 있습니다.

표준 및 호환성

표준	호환성 수준
SQL92	Transact-SQL은 smallint, int, numeric 및 decimal SQL92 정밀 숫자 데이터 유형을 제공합니다. tinyint 유형은 Transact-SQL 확장입니다.

근사 숫자 데이터 유형

기능

산술 연산 동안 반올림을 허용하는 숫자 데이터에 대해 근사 숫자 유형인 float, double precision 및 real을 사용하십시오. 근사 숫자 유형은 넓은 범위의 값을 포함하는 데이터에 특히 적합합니다. 이 유형은 모든 집계 함수(aggregate function) 및 modulo를 제외한 모든 산술 연산을 지원합니다.

근사 숫자 데이터 유형의 이해

부동 소수점 숫자를 저장하기 위해 사용하는 근사 숫자 데이터 유형은 실수를 표현하는 데 있어서 본질적으로 다소 부정확하므로 "근사 숫자"라고 합니다. 이러한 데이터 유형을 사용하려면 이들에 대한 제한 사항을 이해해야 합니다.

부동 소수점 숫자를 출력하거나 표시하면 출력된 표현은 저장된 숫자와 정확히 같지 않으며 저장된 숫자는 사용자가 입력한 숫자와 정확히 같지 않습니다. 대개 저장된 표현은 거의 정확하고 소프트웨어는 인쇄된 출력을 원래 입력한 내용과 같아 보이게 만들지만 부동 소수점 숫자를 계산에 사용하는 경우에는 부정확한 결과가 나올 수도 있음을 이해해야 합니다. 특히 근사 숫자 데이터 유형을 사용하여 반복된 계산을 할 경우 결과는 예상과 상당히 다르고 부정확할 수 있습니다.

부동 소수점 숫자는 컴퓨터에 이진 소수부(즉, 2의 거듭제곱으로 나눈 수)로 저장되는데 우리가 사용하는 숫자는 10진수(10의 거듭제곱)를 사용하기 때문에 부정확성이 발생합니다. 즉, 매우 적은 수의 숫자 집합만 정확하게 저장될 수 있음을 의미합니다. 0.75(3/4)는 이진 소수부(4는 2의 거듭제곱)이기 때문에 정확하게 저장할 수 있지만 0.2(2/10)는 저장할 수 없습니다(10은 2의 거듭제곱이 아님).

일부 숫자에는 정확히 저장하기에 너무 많은 자릿수가 있습니다. double precision은 8개의 이진 바이트로 저장되며 적절한 정밀도를 가진 17자리를 표현할 수 있습니다. real은 4개의 이진 바이트로 저장되며 적절한 정밀도를 가진 6자리를 표현할 수 있습니다.

거의 정확한 숫자로 시작하고 거의 정확한 다른 숫자를 사용하여 계산하는 경우에도 전혀 정확하지 않은 결과를 얻을 수 있습니다. 응용 프로그램에 이런 고려 사항이 중요한 경우 정밀 숫자 데이터 유형을 사용하십시오.

범위, 정밀도(P) 및 저장 장소 크기

real 및 double precision 유형은 운영 체제에서 제공하는 기본 유형입니다. float 유형은 괄호로 묶은 옵션 이진 정밀도(P)를 허용합니다. 1에서 15의 정밀도(P)를 가진 float 열은 real로 저장되며 더 높은 정밀도(P)를 가진 열은 double precision으로 저장됩니다.

세 개의 유형에 대한 범위와 저장 장소 정밀도(P)는 시스템마다 다릅니다.

표 1-10은 각 근사 숫자 유형에 대한 범위와 저장 장소 크기를 보여줍니다. isql은 소수점 다음에 6개의 유효 자릿수만 표시하며 나머지는 반올림합니다.

표 1-10: 근사 숫자 데이터 유형

데이터 유형	저장 장소 바이트
float[(default precision)]	default precision < 16인 경우 4 default precision >= 16인 경우 8
double precision	8
real	4

근사 숫자 데이터 입력

근사 숫자 데이터는 가수 다음에 옵션 지수가 오도록 입력합니다.

- 가수는 기호가 있거나 없는 그리고 소수점이 있거나 없는 숫자입니다. 열의 이진 정밀도(P)는 가수에 허용되는 최대 이진 자릿수를 결정합니다.
- 문자 "e" 또는 "E"로 시작하는 지수는 정수여야 합니다.

엔트리가 나타내는 값은 다음 곱입니다.

```
mantissa * 10EXPONENT
```

예를 들어, 2.4E3은 $2.4 * 10^3$ 또는 2400을 나타냅니다.

Open Client 클라이언트가 입력할 수 있는 값

"NaN"과 "Inf"는 부동 소수 표준으로 각각 "숫자 아님"과 "무한"인 값을 표현하는 데 사용하는 특수한 값입니다. Adaptive Server는 일반적으로 이러한 값을 허용하지 않지만 Open Client 클라이언트는 때로는 이러한 값으로 테이블을 채워 넣을 수 있습니다.

표준 및 호환성

표준	호환성 수준
SQL92	float, double precision 및 real 데이터 유형의 호환 수준은 초급 수준입니다.

통화 데이터 유형

기능

통화 데이터를 저장하려면 money 및 smallmoney 데이터 유형을 사용하십시오. 이러한 유형을 미국 달러와 기타 decimal 통화에 대해 사용할 수 있으나 Adaptive Server에서는 한 가지 통화에서 다른 통화로 변환할 수 없습니다. modulo를 제외한 모든 산술 연산과 모든 집계 함수(aggregate function)에 money 및 smallmoney 데이터를 사용할 수 있습니다.

정밀도

*money*와 *smallmoney* 모두 통화 단위의 1/10,000만큼 정밀하지만 표시할 목적으로 최대 소수점 두 자릿수를 반올림합니다. 디폴트 인쇄 형식은 세 자릿수마다 콤마를 넣습니다.

범위 및 저장 장소 크기

표 1-11은 통화 데이터 유형에 대한 범위와 저장 장소 요구 사항을 요약해서 보여 줍니다.

표 1-11: 통화 데이터 유형

데이터 유형	범위	저장 장소 바이트
money	다음 값 사이의 통화 값: +922,337,203,685,477.5807 및 -922,337,203,685,477.5808	8
smallmoney	다음 값 사이의 통화 값: +214,748.3647 및 -214,748.3648	4

통화 값 입력

E 표기법으로 입력한 통화 값은 float로 해석됩니다. 따라서 엔트리를 money 또는 smallmoney 값으로 저장하면 해당 엔트리가 거부되거나 정밀도(P)가 다소 손실될 수 있습니다.

money 및 smallmoney 값은 달러 기호($), 엔 기호(¥) 또는 파운드 기호(£)와 같은 통화 기호를 앞에 사용하거나 사용하지 않고 입력할 수 있습니다. 음의 값을 입력하려면 통화 기호 다음에 빼기 기호를 넣으십시오. 엔트리에 콤마는 포함시키지 마십시오.

표준 및 호환성

표준	호환성 수준
SQL92	money와 smallmoney 데이터 유형은 Transact-SQL 확장입니다.

타임스탬프 데이터 유형

기능

Client-Library™ 응용 프로그램에서 찾아볼 수 있는 테이블에서는 사용자 정의 timestamp 데이터 유형을 사용하십시오. 자세한 내용은 "찾아보기 모드"를 참고하십시오. Adaptive Server는 행을 수정할 때마다 timestamp 열을 업데이트합니다. 테이블에는 *timestamp* 데이터 유형의 열이 하나만 있을 수 있습니다.

timestamp 열 생성

데이터 유형을 지정하지 않고 timestamp라는 이름의 열을 생성하면 Adaptive Server는 이 열을 timestamp 데이터 유형으로 정의합니다.

```
create table testing
    (c1 int, timestamp, c2 int)
```

timestamp 데이터 유형을 이름이 timestamp인 열에 명시적으로 할당할 수도 있습니다.

```
create table testing
    (c1 int, timestamp timestamp, c2 int)
```

또는 다른 이름을 가진 열에 할당할 수도 있습니다.

```
create table testing
```

```
(c1 int, t_stamp timestamp,c2 int)
```

timestamp라는 열을 생성하고 이것을 다른 데이터 유형에 할당할 수 있습니다. 하지만 다른 사용자에게는 이것이 혼란스러울 수 있고 Open Client™의 browse 함수 또는 tsequal 함수를 사용하지 못할 수 있습니다.

```
create table testing
    (c1 int, timestamp datetime)
```

날짜 및 시간 데이터 유형

기능

절대 날짜와 시간 정보를 저장하려면 datetime 및 smalldatetime 데이터 유형을 사용하십시오. 이진 유형 정보를 저장하려면 timestamp를 사용합니다.

범위 및 저장 장소 요구 사항

표 1-12는 datetime과 smalldatetime 데이터 유형에 대한 범위와 저장 장소 요구 사항을 요약해서 보여 줍니다.

표 1-12: 날짜 및 시간을 저장하는 데 사용하는 Transact-SQL 데이터 유형

데이터 유형	범위	저장 장소 바이트
datetime	1753년 1월 1일부터 9999년 12월 31일까지	8
smalldatetime	1900년 1월 1일부터 2079년 6월 6일까지	4

datetime 및 *smalldatetime* 데이터 입력

datetime 및 *smalldatetime* 데이터 유형은 날짜 부분과 날짜 부분의 앞 또는 뒤에 오는 시간 부분으로 구성됩니다. 날짜나 시간 또는 둘 다 생략할 수 있습니다. datetime과 smalldatetime 값 모두 작은 따옴표 또는 큰 따옴표로 묶어야 합니다.

- datetime 열은 1753년 1월 1일과 9999년 12월 31일 사이의 날짜를 표시합니다. datetime 값은 세부 수준을 지원하는 플랫폼에 한해 1/300초만큼의 정확도를 제공합니다. 저장 장소 크기는 8바이트입니다. 1900년 1월 1일 이후의 일 수에 대해 4바이트, 하루의 시간에 대해 4바이트입니다.
- smalldatetime 열은 1900년 1월 1일부터 2079년 6월 6일까지의 날짜를 분(minute) 정밀도로 표시합니다. 저장 장소 크기는 4바이트입니다. 1900년 1월 1일 이후 일 수에 대해 2바이트, 자정 이후의 분 수에 대해 2바이트입니다.

datetime 또는 smalldatetime 값의 날짜 부분 입력

날짜는 월, 일 및 년도로 구성되며 다양한 형식으로 입력할 수 있습니다.

- 4, 6 또는 8자리의 문자열로 전체 날짜를 구분 없이 입력하거나 각 날짜 부분 사이에 슬래시(/), 하이픈(-) 또는 마침표(.) 구분자를 사용할 수 있습니다.
  - 구분이 없는 문자열로 날짜를 입력할 때에는 해당 문자열 길이에 맞는 형식을 사용하십시오. 한 자릿수의 년도, 월 및 일에 대해서는 앞에 0을 사용하십시오. 잘못된 형식으로 입력된 날짜는 잘못 해석되거나 에러를 일으킬 수 있습니다.
  - 구분자를 사용하여 날짜를 입력할 때에는 set dateformat 옵션을 사용하여 날짜 부분에 예상되는 순서를 결정하십시오. 구분된 문자열의 첫 번째 날짜 부분이 4자리인 경우 Adaptive Server는 문자열을 *yyyy-mm-dd* 형식으로 해석합니다.
- 일부 날짜 형식은 다음과 같이 2자리 년도(*yy*)를 사용합니다.
  - 50 미만의 수는 20*yy*로 해석됩니다. 예를 들어 `01`은 2001, `32`는 2032, `49`는 2049를 나타냅니다.
  - 50 이상의 수는 19*yy*로 해석됩니다. 예를 들어 `50`은 1950, `74`는 1974, 그리고 `99`는 1999를 나타냅니다.
- 월을 숫자나 이름으로 지정할 수 있습니다. 월 이름과 약어는 언어에 따라 다르며 대문자, 소문자 또는 대/소문자를 함께 사용하여 입력할 수 있습니다.
- datetime 또는 smalldatetime 값의 날짜 부분을 생략하면 Adaptive Server는 1900년 1월 1일이라는 디폴트 날짜를 사용합니다.

표 1-13은 datetime 또는 smalldatetime 값의 날짜 부분을 입력하는 데 사용할 수 있는 형식에 대해 설명합니다.

표 1-13: datetime 및 smalldatetime 데이터 유형의 날짜 형식

날짜 형식	설명	예제 엔트리	의미
구분자가 없는 4자리 문자열	*yyyy*로 해석됩니다. 지정된 년도의 1월 1일이 디폴트 날짜입니다.	"1947"	1947년 1월 1일
구분자가 없는 6자리 문자열	*yymmdd*로 해석됩니다. *yy* < 50의 경우 년도는 20*yy*로 해석됩니다. *yy* >= 50의 경우 년도는 19*yy*로 해석됩니다.	"450128" "520128"	2045년 1월 28일 1952년 1월 28일
구분자가 없는 8자리 문자열	*yyyymmdd*로 해석됩니다.	"19940415"	1994년 4월 15일
슬래시, 하이픈, 마침표 또는 위의 조합으로 구분된 2자리의 월, 일 및 년도로 구성된 문자열	dateformat 및 language set 옵션은 날짜 부분에서 예상되는 순서를 결정합니다. us_english의 경우 디폴트 순서는 *mdy*입니다. *yy* < 50의 경우 년도는 20*yy*로 해석됩니다. *yy* >= 50의 경우 년도는 19*yy*로 해석됩니다.	"4/15/94" "4.15.94" "4-15-94" "04.15/94"	모든 엔트리는 dateformat 옵션을 mdy로 설정하면 1994년 4월 15일로 해석됩니다.
슬래시, 하이픈, 마침표 또는 이들의 조합으로 구분된 2자리의 월, 2자리의 일, 4자리의 년도로 구성된 문자열	dateformat 및 language set 옵션은 날짜 부분에서 예상되는 순서를 결정합니다. us_english의 경우 디폴트 순서는 *mdy*입니다.	"04/15.1994"	dateformat 옵션을 mdy로 설정하면 1994년 4월 15일로 해석됩니다.
월의 경우 문자 형식(전체 월 이름이나 표준 약어)으로 입력되며 뒤에 옵션으로 콤마가 올 수 있습니다.	4자리 년도를 입력할 경우 날짜 부분은 순서에 상관없이 입력할 수 있습니다. 일을 생략할 경우 4자리 년도를 모두 지정해야 합니다. 월의 첫 번째 날이 디폴트 날짜입니다. 년도가 2자리(*yy*)인 경우 일 뒤에 년도가 오는 것으로 간주됩니다. *yy* < 50의 경우 년도는 20*yy*로 해석됩니다. *yy* >= 50의 경우 년도는 19*yy*로 해석됩니다.	"April 15, 1994" "1994 15 apr" "1994 April 15" "15 APR 1994" "apr 1994" "mar 16 17" "apr 15 94"	모든 엔트리는 1994년 4월 15일로 해석됩니다. 1994년 4월 1일 2017년 5월 16일 1994년 4월 15일
빈 문자열, " "	1900년 1월 1일이 디폴트 날짜입니다.	" "	1900년 1월 1일

datetime 또는 *smalldatetime* 값의 시간 부분 입력

datetime 또는 smalldatetime 값의 시간 구성 요소는 다음과 같이 지정해야 합니다.

```
hours[:minutes[:seconds[:milliseconds]] [AM | PM]
```

- 자정은 12AM, 정오는 12PM을 사용합니다.
- 시간 값은 콜론 또는 AM이나 PM 식별자를 포함해야 합니다. AM 또는 PM은 대문자, 소문자 또는 대/소문자를 함께 사용하여 입력할 수 있습니다.
- 초 지정은 소수점 뒤에 소수부 또는 콜론 뒤에 밀리초 수를 포함할 수 있습니다. 예를 들어 "12:30:20:1"은 12:30에서 20초와 1밀리초가 지난 것을 의미하며 "12:30:20.1"은 12:30에서 20초와 1/10초가 지난 것을 의미합니다.
- datetime 또는 smalldatetime 값의 시간 부분을 생략하면 Adaptive Server는 디폴트 시간인 12:00:00:000AM을 사용합니다.

datetime 및 *smalldatetime* 값의 표시 형식

datetime 및 smalldatetime 값의 표시 형식은 "Mon dd yyyy hh:mmAM" (또는 "PM")입니다. 예를 들면, "Apr 15 1988 10:23PM"입니다. 초와 밀리초를 표시하고 추가 날짜 스타일과 날짜 부분 순서가 필요면 convert 함수를 사용하여 해당 데이터를 문자열로 변환합니다. Adaptive Server는 밀리초 값을 반올림하거나 truncation할 수 있습니다.

표 1-14는 datetime 엔트리와 표시 값의 몇 가지 예제를 보여 줍니다.

표 1-14: datetime 엔트리의 예제

엔트리	표시된 값
"1947"	Jan 1 1947 12:00AM
"450128 12:30:1PM"	Jan 28 2045 12:30PM
"12:30.1PM 450128"	Jan 28 2045 12:30PM
"14:30.22"	Jan 1 1900 2:30PM
"4am"	Jan 1 1900 4:00AM

패턴이 일치하는 *datetime* 값 찾기

특정 패턴과 일치하는 날짜를 찾으려면 like 키워드를 사용하십시오. 등호 연산자(=)를 사용하여 특정 월, 일, 년도의 datetime 값을 검색하는 경우 Adaptive Server는 시간이 정확히 12:00:00:000AM인 값만 return합니다.

예를 들어, *arrival_time*이라는 열에 값 "9:20"을 삽입하면 Adaptive Server는 이 엔트리를 "Jan 1 1900 9:20AM"으로 변환합니다. 등호 연산자를 사용하여 이 엔트리를 찾으면 발견되지 않습니다.

```
where arrival_time = "9:20" /* does not match */
```

like 연산자를 사용하여 엔트리를 찾을 수 있습니다.

```
where arrival_time like "%9:20%"
```

like를 사용할 때 Adaptive Server는 먼저 날짜를 datetime 형식으로 변환한 후 varchar로 변환합니다. 표시 형식은 현재 언어로 된 3자리 월, 2자리 일, 4자리 년도, 시간과 분, "AM" 또는 "PM" 형식으로 구성됩니다.

like를 사용하여 검색하는 경우 datetime 및 smalldatetime 값의 날짜 부분 입력에 사용할 수 있는 다양한 입력 형식을 사용할 수 없습니다. 표준 표시 형식은 초나 밀리초를 포함하지 않기 때문에 *style* 9 또는 109와 convert 함수를 사용하지 않는 한 like와 일치 패턴을 사용하여 초나 밀리초를 검색할 수 없습니다.

like를 사용하고 있고 월의 일이 1에서 9 사이의 숫자인 경우 datetime 값의 varchar 변환을 일치시키려면 월과 일 사이에 공백 두 개를 삽입합니다. 마찬가지로 시간이 10 미만인 경우 변환 과정에서 년도와 시간 사이에 공백 두 개가 추가됩니다. 구문은 다음과 같습니다.

```
like May 2%
```

위 구문에서 "May"와 "2" 사이에 공백이 하나 있으면 5월 2일은 찾지 않고 5월 20일과 5월 29일 사이의 모든 날짜를 찾습니다. datetime 값은 like 비교 시에만 varchar로 변환되기 때문에 like 비교를 수행할 때에만 추가 공백을 삽입하면 되고 다른 날짜 비교를 수행할 때는 추가 공백을 삽입하지 않아도 됩니다.

날짜 조작

내장된 날짜 함수를 사용하여 datetime 값에서 몇 가지 산술 계산을 할 수 있습니다. "날짜 함수"를 참조하십시오.

표준 및 호환성

표준	호환성 수준
SQL92	datetime 및 smalldatetime 데이터 유형은 Transact-SQL 확장입니다.

문자 데이터 유형

기능

문자, 숫자 및 기호로 구성된 문자열을 저장하려면 문자 데이터 유형을 사용하십시오. us_english와 같은 단일 바이트 문자 집합에 대해서는 고정 길이 데이터 유형인 char(n) 및 unichar(n) 그리고 가변 길이 데이터 유형인 varchar(n) 및 univarchar(n)을 사용합니다. 한국어와 같은 다중 바이트 문자 집합의 경우 고정 길이 데이터 유형 nchar(n)과 가변 길이 데이터 유형 nvarchar(n)을 사용하십시오. 문자 데이터 유형은 최대 255자를 저장할 수 있습니다. 255자보다 긴 문자열에 대해서는 text 데이터 유형을 사용하십시오(text 및 image 데이터 유형에서 설명).

길이 및 저장 장소 크기

고정 길이 데이터 유형 char(n), unichar(n) 및 nchar(n)에 대한 문자 길이를 지정하려면 *n*을 사용하십시오. 할당된 길이보다 짧은 엔트리나 할당된 길이보다 긴 엔트리는 set 명령에서 string_rtruncation 옵션을 on으로 설정하지 않는 한 경고 없이 잘립니다. null을 허용하는 고정 길이 열은 내부적으로 가변 길이 열로 변환됩니다.

가변 길이 데이터 유형인 varchar(n), univarchar(n) 및 nvarchar(n)에 대한 문자 최대 길이를 지정하려면 *n*을 사용하십시오. 가변 길이 열에 있는 데이터의 후미 공백은 잘립니다. 저장 장소 크기는 입력한 데이터의 실제 길이입니다. 가변 길이 변수와 매개변수의 데이터는 모든 후미 공백을 남겨 두지만 정의된 길이로 채우지는 않습니다. 문자 리터럴은 가변 길이 데이터 유형으로 취급됩니다.

고정 길이 열은 가변 길이 열보다 더 많은 저장 공간을 차지하는 경향이 있지만 액세스 속도는 더 빠릅니다. 표 1-15는 여러 문자 데이터 유형의 저장 장소 요구 사항을 요약해서 보여 줍니다.

표 1-15: 문자 데이터 유형

데이터 유형	저장 장소	저장 장소 바이트
char(n)	주민등록 번호나 우편 번호처럼 단일 바이트 문자 집합으로 된 고정 길이 데이터	*n*
unichar(n)	고정 길이 Unicode 데이터, 단일 바이트 문자 집합	*n*@@unicharsize (@@unicharsize equals 2)*
nchar(n)	다중 바이트 문자 집합으로 된 고정 길이 데이터	*n * @@ncharsize*

데이터 유형	저장 장소	저장 장소 바이트
varchar(n)	이름과 같이 단일 바이트 문자 집합으로 된 가변 길이 데이터	입력한 문자의 실제 수
univarchar(n)	가변 길이 Unicode 데이터, 단일 바이트 문자 집합	입력한 문자의 실제 수 * *@@unicharsize*
nvarchar(n)	다중 바이트 문자 집합으로 된 가변 길이 데이터	실제 문자 수* *@@ncharsize*

시스템 함수를 사용하여 열 길이 결정

열 길이를 결정하려면 char_length 문자열 함수와 datalength 시스템 함수를 사용하십시오.

- char_length는 열의 문자 수를 가변 길이 데이터 유형의 후미 공백을 잘라내고 return합니다.
- datalength는 바이트 수를 가변 길이 열에 저장된 데이터의 후미 공백을 잘라내고 return합니다.

char 값이 NULLS를 허용하도록 선언되면 Adaptive Server는 내부적으로 이 값을 varchar로 저장합니다.

min 또는 max 집계 함수(aggregate function)를 char 열에서 사용할 경우 return된 결과는 varchar이므로 모든 후미 공백을 자릅니다.

문자 데이터 입력

문자열은 작은 따옴표 또는 큰 따옴표로 묶어야 합니다. set quoted_identifier on을 사용하는 경우 문자열을 작은 따옴표로 묶으십시오. 그렇지 않으면 Adaptive Server가 문자열을 식별자로 취급합니다.

큰 따옴표 문자가 포함된 문자열은 작은 따옴표로 묶어야 합니다. 작은 따옴표 문자가 포함된 문자열은 큰 따옴표로 묶어야 합니다. 예를 들면 다음과 같습니다.

```
'George said, "There must be a better way."'
"Isn't there a better way?"
```

또는 문자열에 포함하려는 각 따옴표에 대해 두 개의 따옴표를 입력하는 것입니다. 예를 들면 다음과 같습니다.

```
"George said, ""There must be a better way."""
'Isn''t there a better way?'
```

문자열을 화면 다음 줄로 계속하려면 다음 줄로 이동하기 전에 역슬래시(\)를 입력하십시오.

공백 처리

다음 예제는 고정 길이와 가변 길이 문자열을 모두 가진 spaces라는 이름의 테이블을 생성합니다.

```
create table spaces (cnot char(5) not null,
    cnull char(5) null,
    vnot varchar(5) not null,
        vnull varchar(5) null,
    explanation varchar(25) not null)

insert spaces values ("a", "b", "c", "d",
    "pads char-not-null only")
insert spaces values ("1    ", "2    ", "3    ",
    "4    ", "truncates trailing blanks")
insert spaces values ("    e", "    f", "    g",
     "    h", "leading blanks, no change")
insert spaces values ("   w ", "   x ", "   y ",
    "   z ", "truncates trailing blanks")
insert spaces values ("", "", "", "",
    "empty string equals space" )

select "[" + cnot + "]",
       "[" + cnull + "]",
       "[" + vnot + "]",
       "[" + vnull + "]",
    explanation from spaces
        explanation
-------  -------  ------- ------- --------------------
[a    ]  [b]      [c]     [d]     pads char-not-null only
[1    ]  [2]      [3]     [4]     truncates trailing blanks
[    e]  [    f]  [    g] [    h] leading blanks, no change
[   w ]  [   x]   [   y]  [   z]  truncates trailing blanks
[     ]  [ ]      [ ]     [ ]     empty string equals space

(5 rows affected)
```

이 예제는 열의 데이터 유형과 null 유형이 공백을 처리하는 방법을 결정하기 위해 어떻게 상호 작용하는지 보여 줍니다.

- char not null 및 nchar not null 열만이 열의 전체 너비에 채워지고 char null 열은 varchar처럼 취급되며 nchar null 열은 nvarchar처럼 취급됩니다.
- unichar not null 열만 열의 전체 너비에 채워지고 unichar null 열은 univarchar처럼 취급됩니다.
- 선행 공백은 영향을 받지 않습니다.
- char, unichar 및 nchar not null 열 이외에 후미 공백은 truncation됩니다.
- 빈 문자열("")은 하나의 공백으로 취급됩니다. char, *n*char 및 unichar not null 열에서 결과는 공백의 열 길이 필드입니다.

문자 데이터 조작

like 키워드를 사용하여 특정 문자에 대해 문자열을 검색하고 내장 문자열 함수를 사용하여 내용을 조작할 수 있습니다. 숫자로 구성된 문자열은 convert 함수를 사용하여 정밀 숫자 및 근사 숫자 데이터 유형으로 변환한 후에 산술에 사용할 수 있습니다.

표준 및 호환성

표준	호환성 수준
SQL92	Transact-SQL은 char 및 varchar SQL92 데이터 유형을 제공합니다. nchar, nvarchar, unichar 및 univarchar 데이터 유형은 Transact-SQL 확장입니다.

Binary 데이터 유형

기능

그림과 같이 raw 이진 데이터를 16진수와 같은 표기법으로 서버의 논리적 페이지 크기에 대해 최대 열 크기까지 저장하려면 binary(n) 및 varbinary(n) 등과 같은 Binary 데이터 유형을 사용하십시오.

유효한 *binary* 및 *varbinary* 엔트리

이진 데이터는 문자 "0x"로 시작하며 숫자와 대/소문자 A부터 F까지의 모든 조합을 포함할 수 있습니다.

열 길이를 바이트로 지정하거나 1바이트의 디폴트 길이를 사용하려면 *n*을 사용하십시오. 각 바이트는 2개의 이진 자릿수를 저장합니다. *n*보다 긴 값을 입력하면 Adaptive Server는 경고나 에러를 발생시키지 않은 채 엔트리를 지정된 길이로 잘라냅니다.

모든 엔트리의 길이가 거의 같은 데이터에 대해서는 고정 길이 이진 유형인 binary(n)를 사용하십시오.

길이가 각각 다른 데이터에 대해서는 가변 길이 이진 유형인 varbinary(n)를 사용하십시오.

binary 열의 엔트리는 열 길이(*n*)만큼 0으로 채워지기 때문에 varbinary 열보다 많은 저장 공간을 필요로 하지만 액세스되는 속도는 더 빠릅니다.

최대 열 크기를 넘는 엔트리

외부 데이터 페이지에 보다 큰 블록의 이진 데이터(최대 2,147, 483, 647바이트)를 저장하려면 image 데이터 유형을 사용하십시오. 내장 프로시저의 변수나 매개변수에 대해서는 image 데이터 유형을 사용할 수 없습니다. 자세한 내용은 "text 및 image 데이터 유형" 단원을 참조하십시오.

후미 0의 처리

모든 binary not null 열은 열의 전체 너비만큼 0으로 채워집니다. null 값을 허용하는 열은 가변 길이 열로 취급되므로 모든 varbinary 데이터 및 binary null 열에서 후미 0은 잘립니다.

다음 예제에서는 binary 및 varbinary 데이터 유형, NULL 및 NOT NULL의 네 가지 조합을 가진 테이블을 생성합니다. 네 개의 열에 모두 같은 데이터가 삽입되며 열의 데이터 유형에 따라 데이터가 truncation되거나 채워집니다.

```
create table zeros (bnot binary(5) not null,
            bnull binary(5) null,
            vnot varbinary(5) not null,
            vnull varbinary(5) null)
```

```
insert zeros values (0x12345000, 0x12345000,
            0x12345000, 0x12345000)
insert zeros values (0x123, 0x123, 0x123, 0x123)
select * from zeros
bnot              bnull       vnot          vnull
--------------    ---------  ----------     ---------
0x1234500000      0x123450   0x123450       0x123450
0x0123000000      0x0123     0x0123         0x0123
```

각 저장 장소 바이트는 두 개의 이진 자릿수를 가지므로 Adaptive Server는 이진 엔트리가 문자 "0x"와 짝수 개의 자릿수로 구성된 것으로 간주합니다. "0x" 다음에 홀수 개의 자릿수가 있으면 Adaptive Server는 선행 0을 생략한 것으로 간주하고 0을 추가합니다.

입력 값 "0x00"과 "0x0"은 가변 길이 이진 열(binary null, image 및 varbinary 열)에서 "0x00"으로 저장됩니다. 고정 길이 이진 열(binary not null)에서 값은 필드의 전체 길이만큼 0으로 채워집니다.

```
insert zeros values (0x0, 0x0,0x0, 0x0)
select * from zeros where bnot = 0x00
bnot           bnull       vnot        vnull
----------     ------      -----       ------------
0x0000000000   0x00        0x00        0x00
```

입력 값에 "0x"가 포함되어 있지 않으면 Adaptive Server는 해당 값이 ASCII 값인 것으로 간주하고 이를 변환합니다. 예를 들면 다음과 같습니다.

```
create table sample (col_a binary(8))

insert sample values (’002710000000ae1b’)

select * from sample
col_a
------------------
0x3030323731303030
```

플랫폼 종속적

특정 값을 입력하는 정확한 양식은 사용 중인 플랫폼에 따라 다릅니다. 따라서 이진 데이터가 관련된 계산은 시스템마다 다른 결과를 산출합니다.

sum 또는 avg 집계 함수(aggregate function)와 함께 Binary 데이터 유형을 사용할 수 없습니다.

플랫폼 독립적인 16진수 문자열과 정수 간의 변환의 경우 플랫폼 고유의 변환 함수보다는 inttohex 및 hextoint 함수를 사용하십시오. 자세한 내용은 "데이터 유형 변환 함수"를 참조하십시오.

표준 및 호환성

표준	호환성 수준
SQL92	binary 및 varbinary 데이터 유형은 Transact-SQL 확장입니다.

bit 데이터 유형

기능

true/false 및 yes/no 유형의 데이터가 포함된 열에는 bit 데이터 유형을 사용하십시오. syscolumns 시스템 테이블의 status 열은 bit 데이터 유형 열의 고유한 오프셋 위치를 나타냅니다.

bit 열에 데이터 입력

bit 열은 0 또는 1의 값을 갖습니다. 0이나 1 이외의 정수 값도 사용할 수 있지만 항상 1로 해석됩니다.

저장 장소 크기

저장 장소 크기는 1바이트입니다. 테이블에 있는 여러 bit 데이터 유형이 모이면 바이트를 차지합니다. 예를 들어, bit 열이 7개 있으면 1바이트가 되고 *bit* 열이 9개 있으면 2바이트를 차지합니다.

제한

bit 데이터 유형의 열은 NULL이 될 수 없으며 인덱스를 가질 수 없습니다.

표준 및 호환성

표준	호환성 수준
SQL92	Transact-SQL 확장

sysname 데이터 유형

기능

*sysname*은 Adaptive Server 설치 테이프로 배포되는 사용자 정의 데이터 유형으로 시스템 테이블에서 사용됩니다. 정의는 다음과 같습니다.

```
varchar(30) "not null"
```

sysname 데이터 유형 사용

열, 매개변수 또는 변수를 sysname 유형으로 선언할 수 없습니다. 그러나 sysname의 디폴트 유형을 사용하여 사용자 정의 데이터 유형을 생성할 수 있습니다. 그런 다음 사용자 정의 데이터 유형을 사용하여 열, 매개변수 및 변수를 정의할 수 있습니다.

표준 및 호환성

표준	호환성 수준
SQL92	sysname을 비롯하여 모든 사용자 정의 데이터 유형은 Transact-SQL 확장입니다.

text 및 *image* 데이터 유형

기능

text 열은 최대 2,147,483,647 (2^{31}- 1)바이트의 인쇄 가능한 문자를 저장할 수 있는 가변 길이 열입니다.

image 열은 최대 2,147,483,647 (2^{31}- 1)바이트의 16진수와 같은 데이터를 저장할 수 있는 가변 길이 열입니다.

text 또는 *image* 열 정의

text 또는 image 열을 다른 열과 마찬가지로 create table 또는 alter table 구문을 사용하여 정의하십시오. text 및 image 데이터 유형 정의는 길이를 포함하지 않으며 null 값을 허용합니다. 열 정의는 다음 형식을 취합니다.

column_name {text | image} [null]

예를 들어, pubs2 데이터베이스에서 null 값을 허용하는 text 열인 blurb이 있는 blurbs 테이블을 만들기 위한 create table 구문은 다음과 같습니다.

```
create table blurbs
(au_id id not null,
copy text null)
```

pubs2 데이터베이스에서 image 열이 있는 au_pix 테이블을 생성하려면 다음과 같이 합니다.

```
create table au_pix
(au_id          char(11) not null,
pic             image null,
format_type     char(11) null,
bytesize        int null,
pixwidth_hor    char(14) null,
pixwidth_vert   char(14) null)
```

Adaptive Server가 *text* 및 *image* 데이터를 저장하는 방법

Adaptive Server는 테이블의 나머지 부분과 분리된 데이터 페이지의 링크된 목록에 text와 image 데이터를 저장합니다. 각 text나 image 페이지는 최대 1800바이트의 데이터를 저장합니다. 테이블에 대한 모든 text와 image 데이터는 테이블이 포함하고 있는 text와 image 열 수에 관계없이 하나의 page chain에 저장됩니다.

다른 디바이스에 추가 페이지 넣기

sp_placeobject를 사용하여 뒤에 오는 text와 image 데이터 페이지를 다른 논리적 디바이스에 넣을 수 있습니다.

0 채우기

16진수의 홀수 자릿수를 가진 255바이트 이하의 image 값은 선행 0으로 채워집니다("0xaaabb"를 삽입하면 "0x0aaabb"가 됩니다).

데이터 저장 장소에서 partition 작업의 영향

alter table 명령의 partition 옵션을 사용하여 text와 image 열이 들어 있는 테이블을 partition할 수 있습니다. 테이블을 partition하면 테이블에 있는 다른 열에 대해 추가 page chain이 생성되지만 text와 image 열이 저장되는 방식에는 영향을 미치지 *않습니다*.

text 및 *image* 열 초기화

업데이트하거나 null이 아닌 값을 삽입할 때까지 text와 image 열은 초기화되지 않습니다. 초기화되는 동안 null이 아닌 각각의 text 및 image 데이터 값에 적어도 하나의 데이터 페이지가 할당됩니다. 또한 text 및 image 데이터의 위치에 대한 포인터가 테이블에 생성됩니다.

예를 들어, 다음 구문은 testtext 테이블을 생성하고 null이 아닌 값을 삽입하여 blurb 열을 초기화합니다. 이제 열에 유효한 텍스트 포인터가 있고 첫 번째 데이터 페이지가 할당되었습니다.

```
create table texttest
(title_id varchar(6), blurb text null, pub_id
char(4))insert texttest values
("BU7832", "Straight Talk About Computers is an
annotated analysis of what computers can do for you:a
no-hype guide for the critical user.", "1389")
```

다음 구문은 image 값에 대한 테이블을 생성하고 image 열을 초기화합니다.

```
create table imagetest
(image_id varchar(6), imagecol image null, graphic_id
char(4))
insert imagetest values
("94732", 0x0000008300000000000100000000013c, "1389")
```

참고 text 값을 따옴표로 묶고 image 값 앞에 문자 "0x"가 와야 한다는 것을 기억하십시오.

Client-Library 프로그램을 사용하여 text와 image 데이터를 삽입하고 업데이트하는 것에 대한 자세한 내용은 *Client-Library/C Reference Manual*을 참조하십시오.

NULL을 허용하여 공간 절약

빈 text나 image 열에 대해 저장 공간을 절약하려면 null 값을 허용하도록 정의하고 해당 열을 사용할 때까지 null을 삽입하십시오. null 값을 삽입해도 text 또는 image 열은 초기화되지 않으므로 텍스트 포인터를 생성하거나 저장 공간을 할당하지 않습니다. 예를 들어, 다음 구문은 앞에서 만든 testtext 테이블의 title_id와 pub_id 열에 값을 삽입하지만 blurb 텍스트 열은 초기화하지 않습니다.

```
insert texttest
(title_id, pub_id) values ("BU7832", "1389")
```

text나 image 행에 null이 아닌 값을 지정하면 항상 적어도 하나의 데이터 페이지를 포함하게 되며 값을 null로 재설정해도 데이터 페이지의 할당이 해제되지 않습니다.

sysindexes로부터 정보 얻기

text 또는 image 열을 가진 각 테이블에는 sysindexes에 이 열에 대한 정보를 제공하는 추가 행이 있습니다. sysindexes 내의 name 열은 "tablename" 형식을 사용합니다. indid는 항상 255입니다. 이 열은 텍스트 저장소에 대한 정보를 제공합니다.

표 1-16: 텍스트와 이미지 데이터의 저장

열	설명
ioampg	텍스트 page chain의 할당 페이지에 대한 포인터
first	텍스트 데이터의 첫 페이지에 대한 포인터
root	마지막 페이지에 대한 포인터
segment	객체가 들어 있는 세그먼트 번호

이 열에 대한 정보를 보려면 sysindexes 테이블을 쿼리할 수 있습니다. 예를 들어, 다음 쿼리는 pubs2 데이터베이스의 blurbs 테이블이 사용하는 데이터 페이지 수를 보고합니다.

```
select name, data_pgs(object_id("blurbs"), ioampg)
from sysindexes
where name = "tblurbs"
name
------------------------------ -----------
tblurbs                                  7
```

참고 시스템 테이블 게시자는 sysindexes와 systabstats 간의 일대일(1-1) 관계를 보여 줍니다. 이는 정보가 systabstats에 보관되지 않는 text와 image 열의 경우를 제외하고는 옳습니다.

readtext 및 *writetext* 사용

writetext를 사용하여 text 데이터를 입력하거나 readtext를 사용하여 데이터를 읽으려면 우선 text 열을 초기화해야 합니다. 자세한 내용은 readtext와 writetext를 참고하십시오.

update를 사용하여 기존 text와 image 데이터를 NULL로 바꾸면 writetext에서 나중에 사용할 수 있도록 남겨진 첫 페이지를 제외한, 할당된 모든 데이터 페이지가 재사용됩니다. 행에 대한 모든 저장 장소를 할당 해제하려면 delete를 사용하여 전체 행을 제거하십시오.

열이 사용하는 공간 확인

sp_spaceused는 텍스트 데이터가 사용하는 공간에 대한 정보를 index_size로 제공합니다.

```
sp_spaceused blurbs
```

```
name           rowtotal  reserved  data    index_size unused
-------------- --------  --------- ------- ---------- ------
blurbs         6         32 KB     2 KB    16 KB      14 KB
```

text 및 *image* 열의 제한 사항

text 및 image 열은 다음과 같은 용도 및 조건에서는 사용할 수 없습니다.

- 내장 프로시저의 매개변수로 또는 해당 매개변수에 전달되는 값으로
- 로컬 변수
- order by, compute, group by 및 union 구문에서
- 인덱스에서
- 서브쿼리나 조인에서
- like 키워드가 없는 where 절에서
- + 연결 연산자와 함께
- 트리거의 if update 구문에서

text 및 *image* 데이터 선택

다음 전역 변수는 text 및 image 데이터에 관한 정보를 return합니다.

표 1-17: 텍스트 및 이미지 전역 변수

변수	설명
@@textptr	프로세스 중간에 삽입되었거나 업데이트된 마지막 text 또는 image 열의 텍스트 포인터. 이 전역 변수를 textptr() 함수와 혼동하지 마십시오.
@@textcolid	*@@textptr*이 참조하는 열의 ID
@@textdbid	*@@textptr*이 참조하는 열을 가진 객체를 포함하는 데이터베이스 ID
@@textobjid	*@@textptr*이 참조하는 열이 들어 있는 객체의 ID
@@textsize	set textsize 옵션의 현재 값. 이 옵션은 select 구문으로 return되는 *text* 또는 *image* 데이터의 최대 길이를 바이트 단위로 지정합니다. 디폴트는 32K입니다. *@@textsize*의 최대 크기는 2^{31} - 1 (즉, 2,147,483,647)입니다.
@@textts	*@@textptr*이 참조하는 열의 텍스트 타임스탬프

text 및 *image* 데이터 유형 변환

convert 함수를 사용하여 text 값을 char, unichar, varchar 및 univarchar로 변환하고 image 값을 binary 또는 varbinary로 명시적으로 변환할 수 있지만 서버의 논리적 페이지 크기에 대한 최대 열 크기로 결정되는 문자와 Binary 데이터 유형의 최대 크기로 제한됩니다. 길이를 지정하지 않는 경우 변환된 값의 디폴트 길이는 30바이트입니다. implicit conversion은 지원되지 않습니다.

text 데이터의 패턴 일치

text, varchar, univarchar, unichar 또는 char 열에 지정된 패턴이 처음 나타나는 위치를 검색하려면 patindex 함수를 사용하십시오. % 대표 문자가 앞에 오고 그 뒤에 패턴이 와야 합니다(첫 번째나 마지막 문자를 검색할 때는 제외).

특정 패턴을 검색하려면 like 키워드를 사용할 수도 있습니다. 다음 예제에서는 각 패턴 "Net Etiquette"이 포함된 blurbs 테이블의 copy 열에서 text 데이터 값을 선택합니다.

```
select copy from blurbs
where copy like "%Net Etiquette%".
```

중복 행

text 또는 image 데이터에 대한 포인터는 각 행을 고유하게 식별합니다. 따라서 text 또는 image 데이터를 포함하는 테이블은 모든 text 및 image 데이터가 NULL이 아닌 한 중복 행을 포함할 수 없습니다. 중복 행이 있으면 포인터가 초기화되지 않습니다.

표준 및 호환성

표준	호환성 수준
SQL92	text 및 image 데이터 유형은 Transact-SQL 확장입니다.

사용자 정의 데이터 유형

기능

사용자 정의 데이터 유형은 시스템 데이터 유형과 sysname 사용자 정의 데이터 유형을 기초로 생성됩니다. 사용자 정의 데이터 유형을 생성한 후에 이 데이터 유형을 사용하여 열, 매개변수 및 변수를 정의할 수 있습니다. 사용자 정의 데이터 유형으로 생성한 객체는 사용자 정의 데이터 유형의 규칙, 디폴트, null 유형 및 IDENTITY 속성을 상속하고 사용자 정의 데이터 유형이 기초로 하는 시스템 데이터 유형의 디폴트와 null 유형도 상속합니다.

model 데이터베이스에서 자주 사용되는 데이터 유형 생성

사용자 정의 데이터 유형은 그것을 사용할 각 데이터베이스에서 생성해야 합니다. model 데이터베이스에서 자주 사용되는 유형을 만드는 것은 좋은 방법이며 새 데이터베이스를 생성하면 임시 테이블에 사용되는 tempdb를 포함하여 이런 유형이 자동으로 추가됩니다.

사용자 정의 데이터 유형 생성

Adaptive Server에서는 sp_addtype 시스템 프로시저를 사용하여 시스템 데이터 유형을 기초로 하는 사용자 정의 데이터 유형을 생성할 수 있습니다. pubs2 데이터베이스에서 timestamp 또는 tid 데이터 유형과 같은 다른 사용자 정의 데이터 유형을 기초로 하는 사용자 정의 데이터 유형은 생성할 수 없습니다.

sysname 데이터 유형은 이 규칙의 예외입니다. sysname은 사용자 정의 데이터 유형이지만 사용자 정의 데이터 유형을 생성하는 데 사용할 수 있습니다.

사용자 정의 데이터 유형은 데이터베이스 객체입니다. 이름은 대/소문자를 구분하며 식별자 규칙을 준수해야 합니다.

sp_bindrule을 사용하여 사용자 정의 데이터 유형에 규칙을 바인딩하고 sp_bindefault를 사용하여 디폴트를 바인딩할 수 있습니다.

기본적으로 사용자 정의 데이터 유형을 기초로 생성된 객체는 사용자 정의 데이터 유형의 null 유형이나 IDENTITY 속성을 상속합니다. 열 정의에 있는 null 유형이나 IDENTITY 속성은 무시할 수 있습니다.

사용자 정의 데이터 유형 이름 변경

사용자 정의 데이터 유형의 이름을 변경하려면 sp_rename을 사용하십시오.

사용자 정의 데이터 유형 삭제

사용자 정의 데이터 유형을 데이터베이스에서 제거하려면 sp_droptype을 사용하십시오.

참고 테이블에서 이미 사용 중인 데이터 유형은 삭제할 수 없습니다.

데이터 유형에 대한 도움말 보기

시스템 데이터 유형이나 사용자 정의 데이터 유형의 등록정보에 대한 정보를 보려면 sp_help 시스템 프로시저를 사용하십시오. 또한 sp_help를 사용하면 테이블에 있는 각 열의 데이터 유형, 길이, 정밀도(P) 및 소수점 이하 자릿수도 표시할 수 있습니다.

표준 및 호환성

표준	호환성 수준
SQL92	사용자 정의 데이터 유형은 Transact-SQL 확장입니다.

2 장

Transact-SQL 함수

이 장에서는 Transact-SQL 함수에 대해 설명합니다. 함수는 데이터베이스로부터 정보를 return하기 위해 사용되며 select 목록, where 절 그리고 표현식이 허용되는 경우에는 어디서든 사용할 수 있습니다. 또한 내장 프로시저 또는 프로그램의 일부로도 자주 사용됩니다.

함수 유형

표 2-1은 여러 가지 유형의 Transact-SQL 함수 및 각 함수가 return하는 정보 유형을 설명합니다.

표 2-1: Transact-SQL 함수 유형

함수 유형	설명
집계 함수(aggregate function)	쿼리 결과에 새로운 열 또는 추가 행으로 표시되는 합계 값을 생성합니다.
데이터 유형 변환 함수	표현식을 특정 데이터 유형에서 다른 유형으로 변경하고 날짜/시간 정보에 대한 새로운 표시 형식을 지정합니다.
날짜 함수	datetime 및 smalldatetime 값과 이 값들의 구성 요소인 날짜 부분에 대한 계산을 수행합니다.
수학 함수	숫자 데이터에 대한 연산을 수행하는 데 일반적으로 필요한 값을 return합니다.
보안 함수	보안과 관련된 정보를 return합니다.
문자열 함수	이진 데이터, 문자열 및 표현식에 대한 연산을 수행합니다.
시스템 함수	데이터베이스에서 특수한 정보를 return합니다.
텍스트 및 이미지 함수	text 및 image 데이터를 연산하는 데 일반적으로 필요한 값을 입력합니다.

표 2-2에 이러한 함수들이 알파벳순으로 나열되어 있습니다.

표 2-2: Transact-SQL 함수 목록

기능	유형	return 값
abs	수학	표현식의 절대값
acos	수학	코사인으로 지정된 라디안 각도
ascii	문자열	표현식의 첫 문자에 대한 ASCII 코드

기능	유형	return 값
asin	수학	사인으로 지정된 라디안 각도
atan	수학	탄젠트로 지정된 라디안 각도
atn2	수학	사인 및 코사인으로 지정된 라디안 각도
avg	집계	모든 고유 값의 산술 평균
ceiling	수학	지정된 값보다 크거나 같은 최소 정수
char	문자열	정수와 대등한 문자
charindex	문자열	표현식의 시작 위치를 나타내는 정수를 return합니다.
char_length	문자열	표현식의 문자 수
col_length	시스템	정의된 열의 길이
col_name	시스템	테이블 및 ID가 지정된 열의 이름
compare	시스템	선택한 조합 규칙에 따라 다음 값을 return합니다. • 1 − *char_expression1* 값이 *char_expression2* 값보다 크다는 것을 나타냅니다. • 0 − *char_expression1* 값이 *char_expression2* 값과 같다는 것을 나타냅니다. • -1 − *char_expression1* 값이 *char_expression2* 값보다 작다는 것을 나타냅니다.
convert	데이터 유형 변환	다른 데이터 유형 또는 datetime 표시 형식으로 변환된 지정 값
cos	수학	라디안으로 지정된 각도의 코사인
cot	수학	라디안으로 지정된 각도의 코탄젠트
count	집계	null이 아닌 고유한 값의 수
curunreservedpgs	시스템	지정된 디스크 조각의 빈 페이지 수
data_pgs	시스템	지정된 테이블 또는 인덱스에서 사용되는 페이지 수
datalength	시스템	지정된 열 또는 문자열의 실제 바이트 단위 길이
dateadd	날짜	주어진 년도, 분기, 시간 또는 기타 날짜 부분을 지정된 날짜에 더하여 만든 날짜
datediff	날짜	두 날짜 사이의 차이
datename	날짜	datetime 값에서 지정된 부분의 이름
datepart	날짜	datetime 값에서 지정된 부분의 정수 값
db_id	시스템	지정된 데이터베이스의 ID 번호
db_name	시스템	ID 번호가 지정된 데이터베이스의 이름
degrees	수학	지정된 라디안 숫자의 도 단위 각도 크기
difference	문자열	두 개의 soundex 값 사이의 차이
exp	수학	상수 e를 지정된 거듭제곱으로 증가시키는 경우 결과 값
floor	수학	지정된 값보다 작거나 같은 최대 정수
getdate	날짜	시스템의 현재 날짜 및 시간
hextoint	데이터 유형 변환	지정된 16진수 문자열과 대등한 플랫폼 독립 정수

기능	유형	return 값
host_id	시스템	클라이언트 프로세스의 호스트 프로세스 ID
host_name	시스템	클라이언트 프로세스의 현재 호스트 컴퓨터 이름
index_col	시스템	지정된 테이블 또는 뷰에서 인덱스된 열의 이름
inttohex	데이터 유형 변환	지정된 정수와 대등한 플랫폼 독립 16진수
isnull	시스템	*expression1*이 NULL로 계산되는 경우 *expression2* 에 지정된 값을 대체합니다.
is_sec_service_on	보안	보안 서비스를 사용하는 경우 "1", 사용하지 않는 경우 "0"
ltrim	문자열	선두 공백이 제거된 지정 표현식
lct_admin	시스템	마지막 임계값을 관리합니다.
license_enabled	시스템	기능에 대한 라이센스를 사용하는 경우 "1", 사용하지 않는 경우 "0"
log	수학	지정된 숫자의 자연 로그
log10	수학	지정된 숫자의 밑수가 10인 상용 로그
lower	문자열	지정된 표현식과 대등한 대문자
max	집계	열의 최고 값
min	집계	열의 하위 값
mut_excl_roles	시스템	두 롤(role) 사이의 상호 배타성
object_id	시스템	지정된 객체의 객체 ID
object_name	시스템	객체 ID가 지정된 객체의 이름
patindex	문자열, 텍스트 및 이미지	지정된 패턴이 처음 발생한 시작 위치
pi	수학	상수 값 3.1415926535897936
power	수학	지정된 숫자를 주어진 거듭제곱으로 증가시키는 경우 결과 값
proc_role	시스템	프로시저를 실행하기 위한 올바른 롤(role)을 사용자가 가지는 경우 1, 가지지 않는 경우 0
ptn_data_pgs	시스템	Partition에서 사용하는 데이터 페이지 수
radians	수학	지정된 각도의 라디안 단위 각도 크기
rand	수학	지정된 시드 값을 사용하여 생성된 0과 1 사이의 임의 값
replicate	문자열	지정된 횟수만큼 반복되는 표현식으로 구성되는 문자열
reserved_pgs	시스템	지정된 테이블 또는 인덱스에 할당된 페이지 수
reverse	문자열	역순으로 열거된 문자의 지정 문자열
right	문자열	오른쪽에서 지정된 수의 문자를 시작하는 문자 표현식의 일부
role_contain	시스템	*role2*가 *role1*을 포함하는 경우 1
role_id	시스템	사용자에 의해 이름이 지정된 롤(role)의 시스템 롤(role) ID
role_name	시스템	사용자에 의해 시스템 롤(role) ID가 지정된 롤(role)의 이름
round	수학	주어진 소수점 자리로 반올림된 지정된 숫자의 값
rowcnt	시스템	지정된 테이블에 있는 행의 수 추정

기능	유형	return 값
rtrim	문자열	후미 공백이 제거된 지정 표현식
show_role	시스템	로그인에서 현재 사용되는 롤(role)
show_sec_services	보안	사용자가 현재 사용하는 보안 서비스 목록
sign	수학	지정된 값의 부호(양수: +1, 0, 음수: -1)
sin	수학	라디안으로 지정된 각도의 사인
sortkey	시스템	조합 동작에 따라 결과를 정렬하는 데 사용되는 값을 생성합니다. 이를 통해 라틴 문자 기반의 사전식 정렬 순서와 대/소문자 또는 액센트를 구분하는 디폴트 문자 집합과 관계없이 문자 조합 작업을 할 수 있습니다.
soundex	문자열	표현식을 검사하는 방식을 나타내는 4자리 코드
space	문자열	지정된 수의 단일 바이트 공간으로 구성되는 문자열
sqrt	수학	지정된 숫자의 제곱근
str	문자열	지정된 숫자와 대등한 문자
stuff	문자열	특정 문자열에서 지정된 수의 문자를 삭제하고 다른 문자열로 바꾸어 만들어진 문자열
substring	문자열	다른 문자열에서 지정된 수의 문자를 추출하여 만들어진 문자열
sum	집계	값 총계
suser_id	시스템	syslogins 시스템 테이블에 있는 서버 사용자 ID 번호
suser_name	시스템	현재 서버 사용자 또는 사용자 ID가 지정된 사용자의 이름
syb_sendmsg		메시지를 UDP(사용자 데이터그램 프로토콜) 포트로 보냅니다.
tan	수학	라디안으로 지정된 각도의 탄젠트
textptr	텍스트 및 이미지	지정된 text 열의 첫 페이지에 대한 포인터
textvalid	텍스트 및 이미지	지정된 text 열에 대한 포인터가 유효한 경우 1, 유효하지 않은 경우 0
to_unichar	문자열	정수 표현식의 값을 가진 unichar 표현식
tsequal	시스템	찾아보기를 위해 선택된 이후 변경된 행에서 업데이트가 이루어지는 것을 방지하기 위해 timestamp 값을 비교합니다.
uhighsurr	문자열	start 위치의 Unicode 값이 surrogate 쌍(쌍의 처음에 나타남)의 상반부인 경우 1, 아닌 경우 0
ulowsurr	문자열	start 위치의 Unicode 값이 surrogate 쌍(쌍의 두 번째에 나타남)의 하반부인 경우 1, 아닌 경우 0
upper	문자열	지정된 문자열과 대등한 대문자
uscalar	문자열	표현식에서 첫 Unicode 문자의 Unicode 스칼라 값
used_pgs	시스템	지정된 테이블 및 이 테이블의 clustered 인덱스에서 사용하는 페이지 수
user	시스템	현재 서버 사용자의 이름
user_id	시스템	지정된 사용자 또는 현재 사용자의 ID 번호

기능	유형	return 값
user_name	시스템	지정된 사용자 또는 현재 사용자의 데이터베이스에 있는 이름
valid_name	시스템	지정된 문자열의 식별자가 유효하지 않은 경우 0, 유효한 경우 0 이외의 다른 숫자
valid_user	시스템	이 Adaptive Server에 있는 하나 이상의 데이터베이스에서 지정된 ID가 유효한 사용자이거나 가명(alias)인 경우 1

다음 단원에서는 함수 유형에 대해 자세하게 설명합니다. 이 장의 나머지 부분에서는 개별 함수를 알파벳순으로 설명하고 있습니다.

집계 함수 (aggregate function)

집계 함수(aggregate function)는 쿼리 결과에 새로운 열로 표시되는 합계 값을 생성합니다. 집계 함수(aggregate function)는 다음과 같습니다.

- avg
- count
- max
- min
- sum

집계 함수(aggregate function)는 select 목록에서 사용하거나 select 문 또는 서브쿼리의 having 절에서는 사용할 수 있으나 where 절에서는 사용할 수 있습니다.

쿼리에 있는 각 집계에는 자체의 작업 테이블이 필요합니다. 따라서 집계를 사용하는 쿼리는 조회에서 사용할 수 있는 작업 테이블의 최대 수(12)를 초과할 수 없습니다.

집계 함수(aggregate function)를 char 데이터 유형 값에 적용하는 경우 함수는 이 값을 varchar로 묵시적으로 변환하여 모든 후미 공백을 삭제합니다. 마찬가지로 unichar 데이터 유형 값은 univarchar로 묵시적으로 변환됩니다.

max, min 및 count 집계 함수(aggregate function)에는 unichar 데이터 유형을 포함하는 의미가 있습니다.

*group by*와 함께 사용되는 집계

집계는 group by와 함께 자주 사용됩니다. group by를 사용하여 테이블을 그룹으로 구분합니다. 집계는 각 그룹에 대해 단일 값을 생성합니다. group by를 사용하지 않는 경우 테이블에 있는 모든 행 또는 where 절에 의해 정의된 행의 하위 집합에 대해 연산을 수행하면 선택 목록의 집계 함수(aggregate function)는 단일 값을 결과로 생성합니다.

집계 함수(aggregate function)와 NULL 값

집계 함수(aggregate function)는 특정 열에 있는 NULL이 아닌 값의 합계 값을 계산합니다. ansinull 옵션이 off(디폴트)로 설정되어 있으면 집계 함수(aggregate function)가 NULL을 만나는 경우 경고가 발생하지 않습니다. ansinull 옵션이 on으로 설정되어 있으면 집계 함수(aggregate function)가 NULL을 만나는 경우 쿼리는 다음과 같은 SQLSTATE 경고를 return합니다.

```
Warning - null value eliminated in set function
```

벡터 및 스칼라 집계

테이블의 모든 행에 집계 함수(aggregate function)를 사용할 수 있습니다. 이 경우 함수는 스칼라 집계인 단일 값을 생성합니다. 또한 group by 또는 선택적으로 having 절을 사용하여 지정 열 또는 표현식에서 동일한 값을 가지는 모든 행에 집계 함수(aggregate function)를 사용할 수 있습니다. 이 경우 함수는 각 그룹에 대해서 벡터 집계인 단일 값을 생성합니다. 집계 함수(aggregate function)의 결과가 새로운 열에 나타납니다.

스칼라 집계 내에 벡터 집계를 중첩시킬 수 있습니다. 예를 들면 다음과 같습니다.

```
select type, avg(price), avg(avg(price))
from titles
group by type
type
------------  ------------  ------------
```

```
UNDECIDED                NULL              15.23
business                 13.73             15.23
mod_cook                 11.4              15.23
popular_comp             22.95             21.48
psychology               13.50             15.23
trad_cook                14.99             15.96

(6 rows affected)
```

group by 절은 avg(price)와 같은 벡터 집계에 사용됩니다. 스칼라 집계인 avg(avg(price))는 titles 테이블에 있는 유형별 평균 가격들의 평균입니다.

표준 SQL에서 *select_list*에 집계가 포함되는 경우 모든 *select_list* 열은 반드시 집계 함수(aggregate function)를 사용해야 하거나 group by 목록에 존재해야 합니다. 하지만 Transact-SQL에는 이와 같은 제한이 없습니다.

예제 1은 표준 제한을 가지는 select 문을 나타냅니다. 예제 2는 다른 항목(title_id)이 select 목록에 추가된 동일한 구문을 나타냅니다. 또한 표시 내용의 차이를 예로 들어 설명하기 위해 order by가 추가되었습니다. 이와 같은 "추가" 열은 having 절에서도 참조할 수 있습니다.

예제 1

```
select type, avg(price), avg(advance)
from titles
group by type


type
------------  ------------ ------------
UNDECIDED              NULL         NULL
business              13.73     6,281.25
mod_cook              11.49     7,500.00
popular_comp          22.95        21.48
psychology            13.50     4,255.00
trad_cook             14.99        15.96

(6 rows affected)
```

예제 2

```
select type, title_id, avg(price), avg(advance)
from titles
group by type
order by type

type         title_id
-----------  --------    ---------- ---------
```

```
UNDECIDED       MC3026      NULL        NULL
business        BU1032      13.73    6,281.25
business        BU1111      13.73    6,281.25
business        BU2075      13.73    6,281.25
business        BU7832      13.73    6,281.25
mod_cook        MC2222      11.49    7,500.00
mod_cook        MC3021      11.49    7,500.00
popular_comp    PC1035      21.48    7,500.00
popular_comp    PC8888      21.48    7,500.00
popular_comp    PC9999      21.48    7,500.00
psychology      PS1372      13.50    4,255.00
psychology      PS2091      13.50    4,255.00
psychology      PS2106      13.50    4,255.00
psychology      PS3333      13.50    4,255.00
psychology      PS7777      13.50    4,255.00
trad_cook       TC3218      15.96    6,333.33
trad_cook       TC4203      15.96    6,333.33
trad_cook       TC7777      15.96    6,333.33
```

사용자는 group by 다음에 열 이름 또는 임의의 다른 표현식(열 제목 또는 가명(alias) 제외)을 사용할 수 있습니다.

group by 열의 NULL 값은 단일 그룹으로 모아집니다.

select 문에 있는 compute 절은 행 집계를 사용하여 합계 값을 생성합니다. 행 집계를 사용하면 하나의 명령으로 상세 행 및 합계 행을 검색할 수 있습니다. 예제 3은 이 특징을 예로 들어 설명한 것입니다.

예제 3

```
select type, title_id, price, advance
from titles
where type = "psychology"
order by type
compute sum(price), sum(advance) by type

type        title_id    price       advance
----------- -------     ----------  ---------
psychology  PS1372          21.59   7,000.00
psychology  PS2091          10.95   2,275.00
psychology  PS2106          7.00    6,000.00
psychology  PS3333          19.99   2,000.00
psychology  PS7777          7.99    4,000.00
                            sum     sum
                            ------- ----------
                            67.52   21,275.00
```

예제 3과 compute가 없는 예제(예제 1 및 예제 2)들의 차이에 유의하십시오.

sysprocesses 및 syslocks와 같은 가상 테이블에서는 집계 함수(aggregate function)를 사용할 수 없습니다.

커서의 select 절에 집계 함수(aggregate function)가 포함되어 있는 경우 이 커서를 업데이트할 수 없습니다.

행 집계로 사용되는 집계 함수 (aggregate function)

행 집계 함수(aggregate function)는 쿼리 결과에 추가 행으로 나타나는 합계 값을 생성합니다.

집계 함수(aggregate function)를 행 집계로 사용하려면 다음과 같은 구문을 사용합니다.

Start of select statement

compute *row_aggregate*(*column_name*)
[,*row_aggregate*(*column_name*)]...
[by *column_name* [,*column_name*]...]

여기서

- *column_name*은 열 이름이며 괄호 안에 포함되어야 합니다. 정밀 숫자, 근사 숫자 및 money 열에서만 sum 및 avg를 사용할 수 있습니다.

  하나의 compute 절을 사용하여 여러 열에 동일한 함수를 적용할 수 있습니다. 둘 이상의 함수를 사용하는 경우 둘 이상의 compute 절을 사용합니다.

- by는 행 집계 값이 하위 그룹에 대해서 계산됨을 나타냅니다. by 항목의 값이 변경될 때마다 행 집계 값이 생성됩니다. by를 사용하는 경우 반드시 order by를 사용해야 합니다.

  by 다음에 둘 이상의 항목이 나열되면 그룹을 하위 그룹으로 구분하고 각 그룹 수준에서 함수를 적용합니다.

행 집계를 사용하면 하나의 명령으로 상세 행 및 합계 행을 검색할 수 있습니다. 반대로 집계 함수(aggregate function)는 테이블 또는 각 그룹에서 선택된 모든 행에 대해서 일반적으로 단일 값을 생성하며 이 합계 값들이 새로운 열에 표시됩니다.

다음 예제는 이 차이를 보여 주고 있습니다.

```
select type, sum(price), sum(advance)
from titles
```

```
where type like "%cook"
group by type
type
---------- ------------ ----------------
mod_cook          22.98         15,000.00
trad_cook         47.89         19,000.00

(2 rows affected)
select type, price, advance
from titles
where type like "%cook"
order by type
compute sum(price), sum(advance) by type
type        price        advance
---------- ------------  ----------------
mod_cook          2.99          15,000.00
mod_cook          0.00              19.99
             sum          sum
            ----------  ----------------
                 22.98         15,000.00
type       price         advance
---------- ------------  ----------------
trad_cook         11.95          4,000.00
trad_cook         14.99          8,000.00
trad_cook         20.95          7,000.00
             sum              sum
            ----------  ----------------
                 47.89         19,000.00
(7 rows affected)
type       price         advance
---------- ------------  ----------------
mod_cook          2.99           15,000.00
mod_cook          0.00               19.99

Compute Result:
---------------------- -----------------
                 22.98         15,000.00
type       price         advance
---------- ------------  ----------------
trad_cook         11.95          4,000.00
trad_cook         14.99          8,000.00
trad_cook         20.95          7,000.00

Compute Result:
---------------------- -----------------
                 47.89          19,000.00
(7 rows affected)
```

compute 절에 있는 열은 반드시 select 목록에 나타나야 합니다.

select 목록에서의 열 순서는 compute 절에서의 집계 순서를 무시합니다. 예를 들면 다음과 같습니다.

```
create table t1 (a int, b int, c int null)
insert t1 values(1,5,8)
insert t1 values(2,6,9)
(1 row affected)
compute sum(c),  max(b), min(a)
select a, b, c from t1
a            b          c
-----------  ---------- -----------
          1          5          8
          2          6          9

Compute Result:
-----------  ---------- -----------
          1          6          17
```

ansinull 옵션이 off(디폴트)로 설정되어 있으면 행 집계가 null을 만나는 경우 경고가 발생하지 않습니다. ansinull 옵션이 on으로 설정되어 있으면 행 집계가 null을 만나는 경우 쿼리는 다음과 같은 SQLSTATE 경고를 return합니다.

```
Warning - null value eliminated in set function
```

compute 절과 동일한 문에서 select into를 사용할 수 없습니다. compute는 데이터베이스에 저장되지 않는 합계 결과가 포함된 테이블을 생성하기 때문입니다.

데이터 유형 변환 함수

데이터 유형 변환 함수는 특정 데이터 유형의 표현식을 다른 유형으로 변경하고 날짜/시간 정보에 대한 새로운 표시 형식을 지정합니다. 데이터 유형 변환 함수는 다음과 같습니다.

- convert()
- inttohex()
- hextoint()

데이터 유형 변환 함수는 select 목록, where 절 및 표현식이 허용되는 어디서나 사용할 수 있습니다.

Adaptive Server는 특정 유형의 데이터 변환을 자동으로 수행합니다. 이 변환을 *implicit conversion*이라고 합니다. 예를 들어, char 표현식과 datetime 표현식 또는 smallint 표현식 그리고 길이가 다른 int 표현식 또는 char 표현식을 비교하는 경우, Adaptive Server는 데이터의 유형을 다른 유형으로 자동 변환합니다.

내장 데이터 유형 변환 함수 중의 하나를 사용하여 기타 데이터 유형 변환을 명시적으로 요청해야 합니다. 예를 들어, 숫자 표현식을 연결하기 전에 우선 문자 표현식으로 변환해야 합니다.

Adaptive Server는 특정 데이터 유형에 대해서는 다른 특정 데이터 유형으로 명시적 또는 묵시적으로 변환할 수 없게 합니다. 예를 들어, smallint 데이터를 datetime으로 변환하거나 datetime 데이터를 smallint로 변환할 수 없습니다. 지원되지 않는 변환을 수행하면 에러 메시지가 발생합니다.

표 2-3은 개별 데이터 유형 변환이 묵시적으로 또는 명시적으로 수행되는지 또는 지원되지 않는지 여부를 나타냅니다.

표 2-3: explicit/implicit 및 지원되지 않는 데이터 유형 변환

원본 \ 대상	tinyint	smallint	int	decimal	numeric	real	float	{n}char	[n]varchar	unichar	univarchar	text	smallmoney	money	bit	smalldatetime	datetime	binary	varbinary	image
tinyint	–	I	I	I	I	I	I	E	E	E	E	U	I	I	I	U	U	I	I	U
smallint	I	–	I	I	I	I	I	E	E	E	E	U	I	I	I	U	U	I	I	U
int	I	I	–	I	I	I	I	E	E	E	E	U	I	I	I	U	U	I	I	U
decimal	I	I	I	I/E	I/E	I	I	E	E	E	E	U	I	I	I	U	U	I	I	U
numeric	I	I	I	I/E	I/E	I	I	E	E	E	E	U	I	I	I	U	U	I	I	U
real	I	I	I	I	I	–	I	E	E	E	E	U	I	I	I	U	U	I	I	U
float	I	I	I	I	I	I	–	E	E	E	E	U	I	I	I	U	U	I	I	U
[n]char	E	E	E	E	E	E	E	I	I	I	I	I	E	E	E	I	I	I	I	I
[n]varchar	E	E	E	E	E	E	E	I	I	I	I	I	E	E	E	I	I	I	I	I
unichar	E	E	E	E	E	E	E	I	I	–	I	I	E	E	E	E	E	I	I	I
univarchar	E	E	E	E	E	E	E	I	I	I	–	I	E	E	E	E	E	I	I	I
text	U	U	U	U	U	U	U	E	E	E	E	U	U	U	U	U	U	U	U	U
smallmoney	I	I	I	I	I	I	I	I	I	E	E	U	–	I	I	U	U	I	I	U
money	I	I	I	I	I	I	I	I	I	E	E	U	I	–	I	U	U	I	I	U
bit	I	I	I	I	I	I	I	I	I	E	E	U	I	I	–	U	U	I	I	U
smalldatetime	U	U	U	U	U	U	U	I	I	E	E	U	U	U	U	–	I	I	I	U

datetime	U	U	U	U	U	U	U	I	I	E	E	U	U	U	U	I	–	I	I	U
binary	I	I	I	I	I	I	I	I	I	I	I	U	I	I	I	I	I	–	I	I
varbinary	I	I	I	I	I	I	I	I	I	I	I	U	I	I	I	I	I	I	–	I
image	U	U	U	U	U	U	U	U	U	E	E	U	U	U	U	U	U	E	E	U

주석:

E explicit datatype conversion이 필요합니다.

I 묵시적으로 변환을 수행할 수도 있고 데이터 유형 변환 함수를 사용하여 명시적으로 수행할 수도 있습니다.

I/E 정밀도(P)와 소수점 이하 자릿수가 손실되는 경우와 arithabort numeric_truncation이 on으로 설정된 경우에는 explicit datatype conversion을 사용해야 하고 그렇지 않으면 implicit conversion이 허용됩니다.

U 지원되지 않는 변환.

– 데이터 유형 자체로 변환. 이 변환은 사용 가능하지만 아무런 의미가 없습니다.

문자 데이터를 비문자 데이터 유형으로 변환

새로운 유형에 대해서 문자 데이터가 모두 유효한 경우에는 이 데이터를 통화, 날짜/시간, 정밀 숫자, 근사 숫자와 같은 비문자 유형의 데이터로 변환할 수 있습니다. 선두 공백은 무시됩니다. 하지만 하나 이상의 공백으로 구성된 char 표현식을 datetime 표현식으로 변환하는 경우 SQL Server는 이 공백을 디폴트 datetime 값인 "Jan 1, 1900"으로 변환합니다.

허용되지 않은 문자가 데이터에 포함되는 경우 구문 에러가 발생합니다. 다음은 구문 에러를 발생시키는 문자의 예제입니다.

- 정수형 데이터에 있는 콤마 또는 소수점
- 통화 데이터에 있는 콤마
- 정밀 또는 근사 숫자 또는 비트 스트림 데이터에 있는 글자
- 날짜/시간 데이터에 있는 잘못된 월 이름

특정 문자 유형에서 다른 유형으로 변환

다중 바이트의 문자 집합을 단일 바이트의 문자 집합으로 변환하는 경우 하나의 바이트도 없는 문자는 물음표 기호로 전환됩니다.

text 열은 명시적으로 char, nchar, varchar, unichar, univarchar 또는 nvarchar로 변환할 수 있습니다. character 데이터 유형의 최대 길이는 서버의 논리적 페이지 크기에 대한 최대 열 크기로 제한되어 있습니다. 길이를 지정하지 않는 경우 변환된 값의 디폴트 길이는 30바이트입니다.

숫자를 문자 유형으로 변환

정밀 및 근사 숫자 데이터를 문자 유형으로 변환할 수 있습니다. 새로운 데이터 유형의 길이가 짧아 전체 문자열을 수용할 수 없는 경우에는 공간 부족 에러가 발생합니다. 다음은 5자로 된 문자열을 1자로 된 유형으로 변환하는 예를 보여 줍니다.

```
select convert(char(1), 12.34)
Insufficient result space for explicit conversion
of NUMERIC value '12.34' to a CHAR field.
```

참고 float 데이터를 문자 유형으로 변환하는 경우 새로운 데이터 유형의 길이는 적어도 25자 이상이어야 합니다.

통화 유형으로 변환하는 경우 반올림

money 및 smallmoney 유형은 오른쪽 소수점 4자리를 저장하지만 표시를 위해서 소수점 둘째 자리(0.01)로 반올림합니다. 데이터를 통화 유형으로 변환하는 경우 소수점 4자리로 반올림합니다.

통화 유형을 데이터로 변환하는 경우에도 동일한 반올림 동작을 따릅니다. 새로운 데이터 유형이 소수점 3자리 이하 정밀 숫자인 경우 데이터의 소수점 이하 자릿수를 새로운 유형에 일치시킵니다. 예를 들어 다음과 같이 미화 4.50달러를 정수로 변환하면 결과 값은 5입니다.

```
select convert(int, $4.50)
------------
           5
```

money 또는 smallmoney로 변환되는 데이터는 센트와 같은 분수 단위가 아니라 달러와 같은 전체 통화 단위로 가정합니다. 예를 들어, us_english 언어에서 정수 값 5는 5센트가 아니라 5달러의 통화로 변환됩니다.

날짜 / 시간 정보 변환

날짜로 인식될 수 있는 데이터는 datetime 또는 smalldatetime으로 변환됩니다. 잘못된 월 이름은 구문 에러를 발생시킵니다. 데이터 유형의 허용 범위를 초과하는 날짜는 산술 오버플로 에러를 발생시킵니다.

datetime 값을 smalldatetime으로 변환하는 경우 이 값은 가장 가까운 분(minute)으로 반올림됩니다.

숫자 유형 사이의 변환

특정 유형의 숫자 데이터를 다른 유형으로 변환할 수 있습니다. 새로운 데이터 유형의 정밀도(P) 또는 소수점 이하 자릿수가 데이터를 수용하기에 충분하지 않은 경우에는 에러가 발생합니다.

예를 들어, 정수가 필요한 내장 함수에 부동 값 또는 숫자 값을 인자로 입력하는 경우 부동 값 또는 숫자 값은 truncation됩니다. 그러나 Adaptive Server는 분수 부분이 포함된 숫자를 묵시적으로 변환하지 않으며 소수점 이하 자릿수 에러 메시지를 return합니다. 예를 들어, Adaptive Server는 분수 부분이 포함된 숫자에 대해 에러 241을 return하고 다른 데이터 유형이 전달되면 에러 257을 return합니다.

arithabort 및 arithignore 옵션을 사용하여 숫자 변환에서 발생하는 에러를 Adaptive Server가 처리하는 방법을 결정합니다.

참고 arithabort 및 arithignore 옵션은 10.0 이상의 릴리스에서 재정의되었습니다. 응용 프로그램에서 이 옵션을 사용하는 경우 원하는 동작이 제대로 수행되는지 확인하기 위해 이 옵션을 검사합니다.

산술 오버플로 및 0 으로 나눔 에러

Adaptive Server에서 숫자 값을 0으로 나누려는 경우에 0으로 나눔 에러가 발생합니다. 새로운 데이터 유형의 소수점 위치가 적어서 결과를 수용할 수 없는 경우에 산술 오버플로 에러가 발생합니다. 다음과 같은 경우에 산술 오버플로가 발생합니다.

- 보다 낮은 정밀도(P) 또는 소수점 이하 자릿수를 사용하여 정밀 유형으로 명시적 또는 implicit conversion을 수행하는 경우
- 통화 또는 날짜/시간 유형에서 허용되는 범위를 초과하는 데이터를 명시적 또는 묵시적으로 변환하는 경우

- hextoint를 사용하여 4바이트 이상의 저장이 필요한 16진수 문자열 변환을 수행하는 경우

명시적 또는 implicit conversion에 관계없이 산술 오버플로와 0으로 나눔 에러는 심각한 에러로 간주됩니다. arithabort arith_overflow 옵션을 사용하여 Adaptive Server가 이 에러를 처리하는 방법을 결정합니다. 디폴트 설정인 arithabort arith_overflow on은 에러가 발생한 전체 트랜잭션을 롤백(roll back)합니다. 트랜잭션이 포함되지 않은 배치에서 에러가 발생하는 경우 arithabort arith_overflow on은 배치에 있는 이전 명령을 롤백(roll back)하지 않습니다. 또한 Adaptive Server는 배치에서 에러가 발생한 구문 이후에 나오는 구문을 실행하지 않습니다. arithabort arith_overflow off를 설정한 경우 Adaptive Server는 에러가 발생한 구문을 중단하고 트랜잭션 또는 배치에 있는 다른 구문을 계속해서 처리합니다. 구문 결과를 확인하려면 *@@error* 전역 변수를 사용할 수 있습니다.

arithignore arith_overflow 옵션을 사용하여 Adaptive Server가 에러 다음에 메시지를 표시하는지 여부를 결정합니다. 디폴트 설정인 off는 0으로 나눔 에러가 발생하거나 정밀도(P)가 저하되는 경우에 경고 메시지를 표시합니다. arithignore arith_overflow on을 설정하면 에러 다음에 메시지가 표시되지 않도록 합니다. arith_overflow 키워드는 옵션이므로 생략해도 아무런 영향을 미치지 않습니다.

소수점 이하 자릿수 에러

explicit conversion에서 소수점 이하 자릿수 손실이 발생하는 경우 경고 없이 변환 결과가 잘려집니다. 예를 들어 float, numeric 또는 decimal 유형을 integer로 변환하는 경우 Adaptive Server는 결과는 정수로 변환하고 소수점 오른쪽에 있는 모든 숫자는 truncation합니다.

numeric 또는 decimal 유형으로 implicit conversion을 수행하는 경우 소수점 이하 자릿수가 손실되면 소수점 이하 자릿수 에러가 발생합니다. arithabort numeric_truncation 옵션을 사용하여 이와 같은 심각한 에러를 처리하는 방식을 결정합니다. 디폴트 설정인 arithabort numeric_truncation on은 에러를 발생시킨 구문을 중단하고 트랜잭션 또는 배치에 있는 다른 구문을 계속해서 처리합니다. arithabort numeric_truncation off로 설정하는 경우 Adaptive Server는 쿼리 결과를 truncation하고 처리를 계속합니다.

참고 엔트리 수준의 SQL92 호환을 위해서 다음과 같이 설정합니다.

- arithabort arith_overflow off
- arithabort numeric_truncation on
- arithignore off

도메인 에러

함수의 인자가 정의 범위를 초과하는 경우 convert() 함수는 도메인 에러를 발생합니다. 하지만 이 경우는 거의 발생하지 않습니다.

이진 및 정수 유형 사이의 변환

binary 및 varbinary 유형은 "0x" 접두어와 숫자 및 문자열로 구성된 16진수 형식의 데이터를 저장합니다.

이 문자열은 서로 다른 플랫폼에서 다른 방식으로 해석됩니다. 예를 들어, 바이트 0을 most significant로 간주하는 시스템에서 문자열 "0x0000100"은 65536으로 나타나며 바이트 0을 least significant로 간주하는 시스템에서 이 문자열은 256으로 나타납니다.

convert 함수를 사용하여 명시적 또는 묵시적으로 이진 유형을 정수 유형으로 변환할 수 있습니다. 새로운 유형에서 데이터의 길이가 짧은 경우에는 "0x" 접두어를 삭제하고 0으로 채웁니다. 데이터의 길이가 긴 경우에는 데이터를 truncation합니다.

convert 및 implicit datatype conversion은 이진 데이터를 플랫폼에 따라 다르게 평가합니다. 이로 인해 플랫폼에 따라 결과가 다를 수 있습니다. 16진수 문자열을 플랫폼과 관계없이 정수로 변환하려면 hextoint 함수를 사용하고, 정수를 플랫폼과 관계없이 16진수 값으로 변환하려면 inttohex 함수를 사용합니다.

이진 및 숫자 또는 decimal 유형 사이의 변환

binary 및 varbinary 데이터 문자열에서 "0x" 다음에 나오는 첫 두 숫자는 binary 유형을 나타냅니다. "00"은 양수를 나타내고 "01"은 음수를 나타냅니다. binary 또는 varbinary 유형을 numeric 또는 decimal 유형으로 변환하는 경우 "0x" 숫자 다음에 반드시 "00" 또는 "01" 값을 지정해야 합니다. 지정하지 않는 경우 변환에 오류가 발생합니다.

다음은 binary 데이터를 numeric으로 변환하는 예제입니다.

```
select convert(numeric
(38, 18),0x00000000000000006b14bd1e6eea0000000000000000000000000000000000)



----------
123.456000
```

이 예제는 동일한 numeric 데이터를 binary 유형으로 변환합니다.

```
select convert(binary,convert(numeric(38, 18), 123.456))



----------------------------------------------------------------
0x00000000000000006b14bd1e6eea0000000000000000000000000000000000
```

image 열을 binary 유형으로 변환

convert 함수를 사용하여 image 열을 binary 또는 varbinary로 변환할 수 있습니다. binary 데이터 유형의 최대 길이는 서버의 논리적 페이지 크기에 대한 최대 열 크기로 제한되어 있습니다. 길이를 지정하지 않는 경우 변환된 값의 디폴트 길이는 30개의 문자입니다.

다른 유형을 *bit*로 변환

정밀 숫자 및 근사 숫자 유형을 bit 유형으로 묵시적으로 변환할 수 있습니다. 문자 유형에서는 명시적인 convert 함수가 필요합니다.

변환하려는 표현식은 숫자, 소수점, 통화 기호 및 +, - 기호로만 구성되어야 하며 다른 문자가 존재하면 구문 에러가 발생합니다.

0에 해당하는 bit는 0이고, 다른 숫자에 해당하는 bit 는 1입니다.

NULL 값 변환

convert 함수를 사용하여 NULL을 NOT NULL로, NOT NULL을 NULL로 변환할 수 있습니다.

날짜 함수

날짜 함수는 datetime 또는 smalldatetime 데이터 유형의 값을 조작합니다.

날짜 함수는 select 목록 또는 쿼리의 where 절에 사용될 수 있습니다.

1753년 1월 1일 이후의 날짜에 대해서만 datetime 데이터 유형을 사용하십시오. datetime 값은 작은 따옴표 또는 큰 따옴표에 포함되어야 합니다. 이전 날짜에는 char, nchar, varchar 또는 nvarchar를 사용합니다. Adaptive Server는 광범위한 형식의 날짜를 인식합니다. 자세한 내용은 데이터 유형 변환 함수와 "날짜 및 시간 데이터 유형"을 참조하십시오.

Adaptive Server는 문자 값을 datetime 값과 비교하려는 경우와 같이 필요한 경우에 문자와 datetime 값을 자동으로 변환합니다.

날짜 부분

Adaptive Server가 인식하는 날짜 부분의 약어와 허용되는 값은 다음과 같습니다.

날짜 부분	약어	값
year	yy	1753 – 9999(smalldatetime에서는 2079)
quarter	qq	1 – 4
month	mm	1 – 12
week	wk	1 – 54
day	dd	1 – 31
dayofyear	dy	1 – 366
weekday	dw	1 – 7(일요일–토요일)
hour	hh	0 – 23
minute	mi	0 – 59
second	ss	0 – 59
millisecond	ms	0 – 999

두 자리 숫자(*yy*)로 년도를 입력하는 경우

- 50 미만의 수는 20*yy*로 해석됩니다. 예를 들어 `01`은 2001, `32`는 2032, `49`는 2049를 나타냅니다.
- 50 이상의 수는 19*yy*로 해석됩니다. 예를 들어 `50`은 1950, `74`는 1974, `99`는 1999를 나타냅니다.

밀리초 단위 앞에 콜론 또는 마침표를 사용할 수 있습니다. 콜론을 앞에 사용하는 경우 숫자는 천분의 일초를 나타내며 마침표를 앞에 사용하는 경우 한 자리 숫자는 십분의 일초, 두 자리 숫자는 백분의 일초, 세 자리 숫자는 천분의 일초를 나타냅니다. 예를 들어, "12:30:20:1"은 12:30분에서 20초와 천분의 1초가 지난 시간을 나타내며 "12:30:20.1"은 12:30분에서 20초와 십분의 1초가 지난 시간을 나타냅니다. datetime 데이터를 더하는 경우 Adaptive Server는 밀리초 단위의 값을 반올림하거나 truncation할 수 있습니다.

수학 함수

수학 함수는 수학 데이터에 연산을 수행하는 데 일반적으로 필요한 값을 return합니다. 수학 함수 이름은 키워드가 아닙니다.

각 함수는 지정된 유형으로 묵시적으로 변환할 수 있는 인자를 사용합니다. 예를 들어, 근사 숫자 유형을 사용하는 함수는 정수 유형도 사용합니다. Adaptive Server는 인자를 원하는 유형으로 자동 변환합니다.

수학 함수는 다음과 같습니다.

- abs
- acos
- asin
- atan
- atn2
- ceiling
- cos
- cot
- degrees

- exp
- floor
- log
- log10
- pi
- power
- radians
- rand
- round
- sign
- sin
- sqrt
- tan

이 함수들의 도메인 에러 또는 범위 에러를 처리하기 위해 에러 트랩이 제공됩니다. 사용자는 arithabort 및 arithignore 옵션을 설정하여 도메인 에러를 처리하는 방법을 결정합니다.

- arithabort arith_overflow는 0으로 나눔 에러가 발생하거나 정밀도(P)가 손실되는 경우의 다음 동작을 지정합니다. 디폴트 설정인 arithabort arith_overflow on은 전체 트랜잭션을 롤백(roll back)하고 에러가 발생한 배치를 중지합니다. arithabort arith_overflow off로 설정한 경우 Adaptive Server는 에러가 발생한 문을 중지하고 트랜잭션 또는 배치에 있는 다른 문을 계속해서 처리합니다.
- arithabort numeric_truncation은 implicit datatype conversion을 수행하는 동안 정밀 숫자 유형에 의한 소수점 이하 자릿수 손실이 발생한 후에 수행할 동작을 지정합니다. explicit conversion에서 소수점 이하 자릿수가 손실되는 경우 경고가 나타나지 않고 변환 결과가 truncation됩니다. 디폴트 설정인 arithabort numeric_truncation on은 에러를 발생시킨 문을 중단하고 트랜잭션 또는 배치에 있는 다른 구문을 계속해서 처리합니다. arithabort numeric_truncation off로 설정하는 경우 Adaptive Server는 쿼리 결과를 truncation하고 처리를 계속합니다.

- 기본적으로 arithignore arith_overflow 옵션은 off로 설정되므로 Adaptive Server는 숫자 오버플로를 초래하는 임의의 쿼리 다음에 경고 메시지를 표시합니다. arithignore 옵션을 on 으로 설정하면 오버플로 에러를 무시합니다.

  참고 arithabort 및 arithignore 옵션은 10.0 이상의 릴리스에서 재정의 되었습니다. 응용 프로그램에서 이 옵션을 사용하는 경우 원하는 결과가 제대로 나타나는지 확인하기 위해 이 옵션을 검사합니다.

보안 함수

보안 함수는 보안 관련 정보를 return합니다.

보안 함수는 다음과 같습니다.

- is_sec_service_on
- show_sec_services

문자열 함수

문자열 함수는 이진 데이터, 문자열 및 표현식에 대해 연산을 수행합니다. 문자열 함수는 다음과 같습니다.

- ascii
- char
- charindex
- char_length
- difference
- lower
- ltrim
- patindex
- replicate

- reverse
- right
- rtrim
- soundex
- space
- str
- stuff
- substring
- to_unichar
- uhighsurr
- ulowsurr
- upper
- uscalar

문자열 함수는 중첩될 수 있으며 where 절 또는 표현식이 허용되는 어디서나 사용할 수 있습니다. 상수를 문자열 함수에서 사용하는 경우 이 상수를 작은 따옴표 또는 큰 따옴표 안에 포함해야 합니다. 문자열 함수 이름은 키워드가 아닙니다.

각 함수는 지정된 유형으로 묵시적으로 변환할 수 있는 인자를 사용합니다. 예를 들어, 근사 숫자 표현식을 사용하는 함수는 정수 표현식도 사용합니다. Adaptive Server는 인자를 원하는 유형으로 자동 변환합니다.

문자열 함수는 두 가지 문자 표현식을 받아들이지만 유일한 표현식이 unichar인 경우 다른 표현식은 "확장"되고 내부적으로 unichar로 변환됩니다. 이는 혼합 모드 표현식에 대한 기존 규칙을 따릅니다. 하지만 이 변환은 unichar 데이터가 가끔 두 개의 공백을 취하므로 truncation이 일어날 수 있습니다.

문자열 함수의 제한

문자열 함수의 결과는 16K로 제한됩니다.

set string_rtruncation이 on으로 설정된 경우 insert 또는 update를 사용하여 truncation할 경우 에러가 발생합니다. 하지만 *표시된* 문자열이 truncation되었어도 SQL Server는 에러를 기록하지 않습니다. 예를 들면 다음과 같습니다.

```
select replicate("a", 900) + replicate("B", 900)
```

데이터의 첫 16K를 표시하지만 연속적인 데이터는 표시하지 않습니다.

시스템 함수

시스템 함수는 데이터베이스에서 특수한 정보를 return합니다. 시스템 함수는 다음과 같습니다.

- col_length
- col_name
- curunreservedpgs
- data_pgs
- datalength
- db_id
- db_name
- host_id
- host_name
- index_col
- isnull
- lct_admin
- mut_excl_roles
- object_id
- object_name
- proc_role
- ptn_data_pgs
- reserved_pgs
- role_contain
- role_id
- role_name

- rowcnt
- show_role
- suser_id
- suser_name
- tsequal
- used_pgs
- user
- user_id
- user_name
- valid_name
- valid_user

시스템 함수는 select 목록, where 절 그리고 표현식이 허용되는 어디서나 사용할 수 있습니다.

시스템 함수에 대한 인자가 선택적일 경우 현재 데이터베이스, 호스트 컴퓨터, 서버 사용자 또는 데이터베이스 사용자로 간주합니다.

텍스트 및 이미지 함수

텍스트 및 이미지 함수는 text 및 image 데이터에 대해 연산을 수행합니다. 텍스트 및 이미지 함수는 다음과 같습니다.

- textptr
- textvalid

텍스트 및 이미지 내장 함수 이름은 키워드가 아닙니다. set textsize 옵션을 사용하여 select 문이 검색할 수 있는 text 또는 image 데이터의 크기를 제한합니다.

patindex 텍스트 함수는 text 및 image 열에서 사용될 수 있으며 텍스트 및 이미지 함수로 간주될 수도 있습니다.

datalength 함수를 사용하여 text 및 image 열에 있는 데이터의 길이를 구합니다.

text 및 image 열은 다음과 같은 용도 및 조건에서는 사용할 수 없습니다.

- 내장 프로시저에 대한 매개 변수
- 내장 프로시저에 전달되는 값
- 로컬 변수
- order by, compute 및 group by 절에서
- 인덱스에서
- like 키워드가 없는 where 절에서
- 조인에서
- 트리거에서

3 장

함수: *abs – difference*

abs

설명

표현식의 절대값을 return합니다.

구문

abs(*numeric_expression*)

매개변수

numeric_expression
정밀 숫자나 근사 숫자, 통화 유형의 열, 변수, 표현식 또는 이 유형들 중 하나로 변환할 수 있는 모든 데이터 유형으로 표현됩니다.

예제

```
select abs(-1)

-----------
          1
```

절대 값 -1을 return합니다.

사용법

- abs는 수학 함수이며 제공된 표현식의 절대값을 return합니다. 결과 값은 숫자 표현식과 같은 유형으로서 동일한 정밀도(P)와 소수점 이하 자릿수를 갖습니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 abs 함수를 실행할 수 있습니다.

참조

함수 – ceiling, floor, round, sign

acos

설명

코사인이 지정된 각도(라디안 단위)를 return합니다.

구문

acos(*cosine*)

매개변수

cosine
코사인 각도이며 float, real, double precision 유형의 열 이름, 변수, 상수 또는 이 유형들 중 하나로 묵시적으로 변환할 수 있는 모든 데이터 유형으로 표현됩니다.

예제

```
select acos(0.52)

--------------------
             1.023945
```

코사인 각도가 0.52인 각도를 return합니다.

사용법

- acos는 수학 함수이며 코사인이 지정된 값이 되는 각도(라디안 단위)를 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 acos 함수를 실행할 수 있습니다.

참조

함수 – cos, degrees, radians

ascii

설명

표현식에서 첫 문자에 대한 ASCII 코드를 return합니다.

구문

ascii(*char_expr* | *uchar_expr*)

매개변수

char_expr
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형 등의 상수 표현식입니다.

예제

```
select au_lname, ascii(au_lname) from authors
where ascii(au_lname) < 70

 au_lname
 ------------------------------ -----------
 Bennet                                  66
 Blotchet-Halls                          66
 Carson                                  67
 DeFrance                                68
 Dull                                    68
```

ASCII 코드가 70보다 작으면 저자의 마지막 이름과 마지막 이름의 첫 글자에 해당하는 ACSII 코드를 return합니다.

사용법

- ascii는 문자열 함수이며 표현식에서 첫 문자에 해당하는 ASCII 코드를 return합니다.
- 문자열 함수는 두 가지 문자 표현식을 받아들이지만 한 개의 표현식만이 unichar인 경우 다른 표현식은 "확장"되고 내부적으로 unichar로 변환됩니다. 이는 혼합 모드 표현식에 대한 기존 규칙을 따릅니다. 하지만 이 변환은 unichar 데이터가 경우에 따라서는 공백을 두 번 취하므로 truncation이 일어날 수 있습니다.
- *char_expr* 또는 *uchar_expr* 함수가 NULL이면 NULL을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 ascii 함수를 실행할 수 있습니다.

참조 *함수* – char, to_unichar

asin

설명 사인이 지정되어 있는 각도(라디안 단위)를 return합니다.

구문 asin(*sine*)

매개변수

sine

사인 각도이며 float, real, double precision 유형의 열 이름, 변수, 상수 또는 이 유형들 중 하나로 묵시적으로 변환할 수 있는 모든 데이터 유형으로 표현됩니다.

예제

```
select asin(0.52)

--------------------
                0.546851
```

사용법

- asin은 수학 함수이며 사인이 지정된 값이 되는 각도(라디안 단위)를 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 asin 함수를 실행할 수 있습니다.

참조 *함수* – degrees, radians, sin

atan

설명 탄젠트가 지정되어 있는 각도(라디안 단위)를 return합니다.

구문 atan(*tangent*)

매개변수

tangent
탄젠트 각도이며 float, real, double precision 유형의 열 이름, 변수, 상수 또는 이 유형들 중 하나로 변환할 수 있는 모든 데이터 유형으로 표현됩니다.

예제

```
select atan(0.50)

--------------------
            0.463648
```

사용법

- atan은 수학 함수이며 탄젠트가 지정된 값이 되는 각도(라디안 단위)를 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 atan 함수를 실행할 수 있습니다.

참조 *함수* – atn2, degrees, radians, tan

atn2

설명 사인과 코사인이 지정되어 있는 각도(라디안 단위)를 return합니다.

구문 atn2(*sine*, *cosine*)

매개변수

sine
사인 각도이며 float, real, double precision 유형의 열 이름, 변수, 상수 또는 이 유형들 중 하나로 묵시적으로 변환할 수 있는 모든 데이터 유형으로 표현됩니다.

cosine
코사인 각도이며 float, real, double precision 유형의 열 이름, 변수, 상수 또는 이 유형들 중 하나로 묵시적으로 변환할 수 있는 모든 데이터 유형으로 표현됩니다.

예제

```
select atn2(.50, .48)

--------------------
            0.805803
```

사용법

- atn2는 수학 함수이며 사인과 코사인이 지정되어 있는 각도(라디안 단위)를 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 atn2 함수를 실행할 수 있습니다.

참조

함수 – atan, degrees, radians, tan

avg

설명

모든(구별된) 값의 숫자 평균을 return합니다.

구문

avg([all | distinct] *expression*)

매개변수

all
: avg를 모든 값에 적용합니다. all이 디폴트입니다.

distinct
: avg를 적용하기 전에 중복되는 값을 제거합니다. distinct는 선택 사항입니다.

expression
: 열 이름, 상수, 함수, 산술 또는 비트 단위 연산자나 서브쿼리로 연결된 열 이름, 상수, 함수의 모든 조합입니다. 집계에서 표현식은 열 이름인 경우가 많습니다. 자세한 내용은 179 페이지의 "표현식"을 참조하십시오.

예제

예제 1

```
select avg(advance), sum(total_sales)
from titles
where type = "business"


----------------------- -----------
               6,281.25       30788
```

모든 비즈니스 관련 도서의 총 매출 합계와 평균 선금을 계산합니다. 이들 집계 함수(aggregate function) 각각은 검색한 모든 행을 대상으로 계산한 하나의 합계 값을 산출합니다.

예제 2

```
select type, avg(advance), sum(total_sales)
from titles
group by type


type
 ------------ ----------------------- -----------
 UNDECIDED                      NULL        NULL
 business                   6,281.25       30788
 mod_cook                   7,500.00       24278
 popular_comp               7,500.00       12875
 psychology                 4,255.00        9939
 trad_cook                  6,333.33       19566
```

group by 절과 함께 사용하면 집계 함수(aggregate function)는 전체 테이블이 아니라 각 그룹에 대해서 단일 값을 산출합니다. 이 구문은 각 유형의 도서에 대한 합계 값을 산출합니다.

예제 3

```
select pub_id, sum(advance), avg(price)
from titles
group by pub_id
having sum(advance) > $25000 and avg(price) > $15
```

titles 테이블을 출판사 별로 그룹화한 후 총 선금 지급이 $25,000을 넘고 도서의 평균 가격이 $15를 넘는 출판사 그룹만 포함시킵니다.

```
pub_id
------ -------------------- --------------------
0877              41,000.00                15.41
1389              30,000.00                18.98
```

사용법

- avg는 집계 함수(aggregate function)이며 한 열에 있는 모든 값의 평균을 구합니다. avg는 숫자(integer, floating point 또는 money) 데이터 유형에만 사용할 수 있습니다. 평균 계산에서 NULL 값은 무시됩니다.
- 집계 함수(aggregate function)에 대한 일반적인 사항은 45 페이지의 "집계 함수(aggregate function)"를 참조하십시오.
- 정수 데이터의 평균을 계산할 때 해당 열의 데이터 유형이 smallint 또는 tinyint인 경우에도 Adaptive Server는 이를 int 값으로 처리합니다. DB-Library 프로그램에서 오버플로 에러를 피하려면 평균 또는 합계의 결과 값에 대한 모든 변수를 int 유형으로 선언하십시오.

- avg()는 Binary 데이터 유형에는 사용할 수 없습니다.
- 평균 값은 숫자 데이터 유형에 대해서만 정의되므로 Unicode 표현식에 사용하면 에러가 발생합니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 avg 함수를 실행할 수 있습니다.

참조

함수 – max, min

ceiling

설명

지정된 값보다 크거나 같은 최소 정수를 return합니다.

구문

ceiling(*value*)

매개변수

value

열 이름, 변수 또는 정밀 숫자, 근사 숫자, 통화 등의 유형 또는 이 유형들 중 하나로 묵시적으로 변환될 수 있는 모든 데이터 유형의 표현식입니다.

예제

예제 1

```
select ceiling(123.45)

124
```

예제 2

```
select ceiling(-123.45)

-123
```

예제 3

```
select ceiling(1.2345E2)

24.000000
```

예제 4

```
select ceiling(-1.2345E2)

-123.000000
```

예제 5

```
select ceiling($123.45)

124.00
```

예제 6

```
select discount, ceiling(discount) from salesdetail
where title_id = "PS3333"

discount
 -------------------- --------------------
            45.000000            45.000000
            46.700000            47.000000
            46.700000            47.000000
            50.000000            50.000000
```

사용법

- ceiling은 수학 함수이며 지정된 값보다 크거나 같은 최소 정수를 return합니다. return 값은 지정된 값과 동일한 데이터 유형을 갖습니다.

  numeric 및 decimal 값의 경우 결과 값이 제공된 값과 같고 (0)의 소수점 이하 자릿수를 갖습니다.

- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 ceiling 함수를 실행할 수 있습니다.

참조

명령 – set

함수 – abs, floor, round, sign

char

설명

정수에 해당하는 문자를 return합니다.

구문

char(*integer_expr*)

매개변수

integer_expr
0과 255 사이의 모든 정수(tinyint, smallint,, int등) 열 이름, 변수 또는 상수 표현식입니다.

예제

예제 1

```
select char(42)

-
*
```

예제 2

```
select xxx = char(65)

xxx
---
A
```

사용법

- char는 문자열 함수이며 단일 바이트 정수 값을 문자 값으로 변환합니다. char는 주로 ascii의 역함수로 사용됩니다.
- char는 char 데이터 유형을 return합니다. 결과 값이 다중 바이트 문자의 첫 바이트이면 해당 문자가 정의되지 않을 수 있습니다.
- *char_expr*이 NULL이면 NULL을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

char를 사용하여 출력 서식 다시 지정

- 연속 및 char() 값을 사용하여 탭이나 줄 바꿈 표시를 추가함으로써 출력 결과의 서식을 다시 지정할 수 있습니다. char(10)은 줄 바꿈 표시로 char(9)는 탭으로 변환됩니다.

  예를 들어 다음과 같습니다.

```
/* just a space */
select title_id +  " " + title from titles where title_id = "T67061"
/* a return */
select title_id + char(10) + title from titles where title_id = "T67061"
/* a tab */
select title_id + char(9) + title from titles where title_id = "T67061"


------------------------------------------------------------------------
T67061 Programming with Curses
------------------------------------------------------------------------
T67061


Programming with Curses
------------------------------------------------------------------------
T67061      Programming with Curses
```

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 char 함수를 실행할 수 있습니다.

참조 *함수* – ascii, str

charindex

설명 표현식의 시작 위치를 나타내는 정수를 return합니다.

구문 charindex(*expression1*, *expression2*)

매개변수 *expression*
이진 또는 문자 열 이름, 변수 또는 상수 표현식입니다. char, varchar, nchar, nvarchar, unichar 또는 univarchar 데이터, binary 또는 varbinary가 될 수 있습니다.

예제

```
select charindex("wonderful", notes)
from titles
where title_id = "TC3218"

-----------
        46
```

titles 테이블의 notes 열에서 문자 표현식이 "wonderful"로 시작되는 위치를 return합니다.

사용법

- charindex는 문자열 함수이며 *expression1*이 처음 발생하는 경우에 대해 *expression2*를 검색하고 시작 위치를 나타내는 정수를 return합니다. *expression1*을 찾을 수 없으면 charindex가 0을 return합니다.
- *expression1*에 대표 문자가 들어 있으면 charindex에서 이를 리터럴로 다룹니다.
- *char_expr* 또는 *uchar_expr*이 NULL이면 NULL을 return합니다.
- varchar 표현식이 하나의 매개변수로 주어지고 unichar 표현식이 다른 매개변수로 주어진 경우 varchar 표현식은 묵시적으로 unichar로 변환되며 일부 truncation될 수 있습니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 charindex 함수를 실행할 수 있습니다.

참조 *함수* – patindex

char_length

설명 표현식에서 문자의 개수를 return합니다.

구문

char_length(*char_expr*|*uchar_expr*)

매개변수

char_expr
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형의 상수 표현식입니다.

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형 등의 상수 표현식입니다.

예제

예제 1

```
select char_length(notes) from titles
where title_id = "PC9999"

 -----------
          39
```

예제 2

```
declare @var1 varchar(20), @var2 varchar(20), @char
char(20)
select @var1 = "abcd", @var2 = "abcd    ",
        @char = "abcd"
select char_length(@var1), char_length(@var2),
char_length(@char)

 ----------- ----------- -----------
           4           8          20
```

사용법

- char_length는 문자열 함수이며 텍스트 값이나 문자 표현식에서 문자의 개수를 나타내는 정수를 return합니다.
- 가변 길이 열과 변수에 대해서는 char_length가 문자의 개수(해당 열이나 변수에 정의되어 있는 길이가 아님)를 return합니다. 명시적인 후미 공백이 가변 길이 변수에 포함되어 있는 경우에는 삭제되지 않습니다. 리터럴 및 고정 길이 문자 열과 변수에 대해서는 char_length가 후미 공백의 표현식을 삭제하지 않습니다(예제 2 참고).
- 다중 바이트 문자 집합에 대해서는 해당 표현식의 문자 개수가 바이트 수보다 적은 경우가 대부분입니다. 바이트 수를 확인하려면 datalength를 사용하십시오.
- Unicode 표현식의 경우에는 표현식에서 Unicode 값(바이트가 아님)의 수를 return합니다. surrogate 쌍은 두 개의 Unicode 값으로 계산됩니다.
- *char_expr* 또는 *uchar_expr*이 NULL이면 char_length는 NULL을 return합니다.

- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 char_length 함수를 실행할 수 있습니다.

참조 *함수* – datalength

col_length

설명 정의되어 있는 열의 길이를 return합니다.

구문 col_length(*object_name*, *column_name*)

매개변수

object_name
테이블, 뷰, 프로시저, 트리거, 디폴트 또는 규칙 등과 같은 데이터베이스 객체의 이름입니다. 완전히 한정된 이름(즉, 데이터베이스와 소유자 이름을 포함할 수 있음)으로 지정할 수 있습니다. 이때 이름은 따옴표로 묶어야 합니다.

column_name
열의 이름입니다.

예제

```
select x = col_length("titles", "title")

 x
 ----
    80
```

titles 테이블에서 title 열의 길이를 구합니다. "x"는 결과에 열 제목을 부여합니다.

사용법

- col_length는 시스템 함수이며 정의된 열 길이를 return합니다.
- 시스템 함수에 대한 일반적인 사항은 64 페이지의 "시스템 함수"를 참조하십시오.
- 각 행에 저장되어 있는 데이터의 실제 길이를 구하려면 datalength를 사용하십시오.
- text 및 image 열의 경우 col_length는 실제 텍스트 페이지에 대한 binary(16) 포인터의 길이인 16을 return합니다.
- unichar 열의 경우 정의된 길이는 열이 정의될 때 선언된 Unicode 값의 수입니다(바이트 수가 아님).

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 col_length 함수를 실행할 수 있습니다.

참조 *함수* – datalength

col_name

설명 테이블과 열 ID가 지정되어 있는 열의 이름을 return합니다.

구문 col_name(*object_id*, *column_id*[, *database_id*])

매개변수

object_id
테이블, 뷰 또는 기타 데이터베이스 객체에 대한 객체 ID인 숫자 표현식입니다. 객체 ID는 sysobjects의 id 열에 저장됩니다.

column_id
열의 열 ID인 숫자 표현식입니다. 이들은 syscolumns의 colid 열에 저장되어 있습니다.

database_id
데이터베이스의 ID인 숫자 표현식입니다. 이들은 sysdatabases의 db_id 열에 저장되어 있습니다.

예제

```
select col_name(208003772, 2)

------------------------------
title
```

사용법

- col_name은 시스템 함수이며 열의 이름을 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 col_name 함수를 실행할 수 있습니다.

참조 *함수* – db_id, object_id

compare

설명 사용자가 대체 조합 규칙에 따라 두 문자열을 직접 비교할 수 있도록 합니다.

구문

compare (*char_expression1*|u*char_expression1),*
(char_expression2|*uchar_expression2)*
[,{*collation_name* | *collation_ID*}])

매개변수

char_expression1 또는 uchar_expression 1
char_expression2 또는 *uchar_expression 2*에 비교될 문자 표현식입니다.

char_expression2 또는 *uchar_expression2*
char_expression1 또는 *uchar_expression1*을 비교할 문자 표현식입니다.

*char_expression1*과 *char_expression2*는 다음 중 하나입니다.

- 문자 유형(char, varchar, nchar, nvarchar)
- 문자 변수
- 작은 따옴표나 큰 따옴표로 묶은 상수 문자 표현식

*uchar_expression1*과 *uchar_expression2*는 다음 중 하나일 수 있습니다.

- 문자 유형(unichar 또는 univarchar)
- 문자 변수
- 작은 따옴표나 큰 따옴표로 묶은 상수 문자 표현식

collation_name
사용할 조합을 지정하는 인용 문자열이나 문자 변수입니다. 표 3-1은 유효한 값을 보여 줍니다.

collation_ID
사용할 조합을 지정하는 정수 형태의 상수나 변수입니다. 표 3-1은 유효한 값을 보여 줍니다.

사용법

- compare 함수는 선택한 조합 규칙에 따라 다음 값을 return합니다.
  - 1 − *char_expression1* 값 또는 *uchar_expression1* 값이 *char_expression2* 값 또는 *uchar_expression2* 값보다 크다는 것을 나타냅니다.
  - 0 − *char_expression1* 값 또는 *uchar_expression1* 값이 *char_expression2* 값 또는 *uchar_expression2* 값과 같다는 것을 보여 줍니다.
  - -1 − *char_expression1* 값 또는 *uchar_expression1* 값이 *char_expression2* 값 또는 *uchar expression2* 값보다 작다는 것을 나타냅니다.

- *char_expression1*, *uchar_expression1*과 *char_expression2*, *uchar_expression2*는 모두 서버의 디폴트 문자 집합으로 인코딩된 문자여야 합니다.
- *char_expression1*, *uchar_expression 1* 또는 *char_expression2*, *uchar_expression2* 중 하나 또는 모두가 빈 문자열일 수 있습니다.
  - *char_expression2* 또는 *uchar_expression2*가 비어 있는 경우 함수는 1을 return합니다.
  - 두 문자열이 모두 빈 문자열이면 서로 같은 것으로 간주되어 함수는 0 값을 return합니다.
  - *char_expression1* 또는 *uchar_expression 1*이 비어 있는 경우 함수는 -1을 return합니다.

  compare 함수는 빈 문자열과 공백만으로 이루어진 문자열을 동등하게 취급하지 않습니다. compare는 sortkey 함수를 사용해서 비교를 위한 조합 키를 생성합니다. 그러므로 완전히 빈 문자열, 하나의 공백을 포함한 문자열, 두 개의 공백을 포함한 문자열은 동등하게 비교되지 않습니다.
- *char_expression1*, *uchar_expression1* 또는 *char_expression2*, *uchar_expression2*가 NULL이면 결과도 NULL이 됩니다.
- varchar 표현식이 하나의 매개변수로 주어지고 unichar 표현식이 다른 매개변수로 주어진 경우 varchar 표현식은 묵시적으로 unichar 표현식으로 변환되며 일부 truncation될 수 있습니다.
- *collation_name* 또는 *collation_ID*의 값을 지정하지 않으면 compare는 이진 조합으로 가정합니다.
- 표 3-1에서는 *collation_name* 및 *collation_ID*에 대해 유효한 값을 나열합니다.

표 3-1: 조합 이름과 ID

설명	조합 이름	조합 ID
이진 정렬	binary	50
디폴트 Unicode 다국어	default	0
CP 850 대체, 액센트 비구분	altnoacc	39
CP 850 대체, 소문자 우선	altdict	45
CP 850 대체, 대/소문자 비구분	altnocsp	46
CP 850 스칸디나비아어 사전	scandict	47
CP 850 스칸디나비아어, 대/소문자 비구문	scannocp	48
GB 병음	gbpinyin	해당 사항 없음
Latin-1 영어, 불어, 독일어 사전	dict	51

설명	조합 이름	조합 ID
Latin-1 영어, 불어, 독일어, 대/소문자 비구분	nocase	52
Latin-1 영어, 불어, 독일어, 대/소문자 비구분	nocasep	53
Latin-1 영어, 불어, 독일어, 액센트 비구분	noaccent	54
Latin-1 스페인어 사전	espdict	55
Latin-1 스페인어, 대/소문자 비구분	espnocs	56
Latin-1 스페인어, 액센트 비구분	espnoac	57
ISO 8859-5 키릴어 사전	cyrdict	63
ISO 8859-5 러시아어 사전	rusdict	58
ISO 8859-9 터키어 사전	turdict	72
Shift-JIS binary 순서	sjisbin	259
태국어 사전	thaidict	1

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 compare 함수를 실행할 수 있습니다.

참조 *함수* – sortkey

convert

설명 다른 데이터 유형 또는 다른 datetime 표시 형식으로 변환되도록 지정된 값을 return합니다.

구문

convert (*datatype* [(*length*) | (*precision*[, *scale*])]
[null | not null], *expression* [, *style*])

매개변수

datatype

표현식을 변환하기 위한 시스템 제공 데이터 유형(예를 들어, char (10), unichar (10), varbinary (50) 또는 int)입니다. 사용자 정의 데이터 유형을 사용할 수 없습니다.

데이터베이스에서 Java를 사용할 수 있을 경우 *datatype*도 현재 데이터베이스에서 Java-SQL 클래스가 될 수 있습니다.

length

char, nchar, unichar, univarchar, varchar, nvarchar, binary 및 varbinary 등의 데이터 유형과 함께 사용하는 선택적 매개변수입니다. 길이를 입력하지 않으면 Adaptive Server가 이진 유형에 대해 30바이트와 문자 유형에 대해 30자까지 데이터를 truncation합니다. 문자와 이진 표현식에 사용할 수 있는 최대 길이는 64K입니다.

precision

numeric 또는 decimal 데이터 유형에서 중요한 자리의 수입니다. float 데이터 유형의 경우 precision은 가수에서 중요한 이진 자릿수가 됩니다. 따로 precision을 입력하지 않으면 Adaptive Server에서는 numeric 및 decimal 데이터 유형에 대해 디폴트 정밀도(P) 18을 사용합니다.

scale

numeric 또는 decimal 데이터 유형에서 소수점의 오른쪽에 있는 숫자의 개수입니다. 따로 scale을 입력하지 않으면 Adaptive Server에서는 디폴트로 0을 사용합니다.

null | not null

결과 표현식의 null 값 허용 여부를 지정합니다. null 또는 not null 중 하나를 입력하지 않으면 변환된 결과에도 표현식과 동일한 내용이 나타납니다.

expression

데이터 유형이나 날짜 형식이 다른 유형이나 형식으로 변환되는 값입니다.

데이터베이스에서 Java를 사용할 수 있을 경우 *expression*은 Java-SQL 클래스로 변환되는 값이 될 수 있습니다.

Unichar를 대상 데이터 유형으로 사용하면 디폴트 길이인 30 Unicode 값은 길이가 지정되지 않은 경우 사용됩니다.

style

변환된 데이터에 사용하는 표시 형식입니다. money 또는 smallmoney 데이터를 문자 유형으로 변환할 때에는 1이라는 *style*을 사용하여 세 자릿수마다 콤마를 표시하십시오.

datetime 또는 smalldatetime 데이터를 문자 유형으로 변환할 때에는 표 3-2에서 style 개수를 사용하여 표시 형식을 지정하십시오. 가장 왼쪽 열에 나타나 있는 값은 두 자리 년도(yy)를 표시합니다. 네 자리 년도(yyyy)의 경우에는 100을 추가하거나 중간 열에 있는 값을 사용하십시오.

표 3-2: 날짜/시간 정보에 대한 표시 형식

세기 표시 안함(yy)	세기 표시함(yyyy)	결과
해당 사항 없음	0 또는 100	*mon dd yyyy hh:mi*AM(또는 PM)
1	101	*mm/dd/yy*
2	102	*yy.mm.dd*
3	103	*dd/mm/yy*
4	104	*dd.mm.yy*
5	105	*dd-mm-yy*

세기 표시 안함(yy)	세기 표시함(yyyy)	결과
6	106	*dd mon yy*
7	107	*mon dd, yy*
8	108	*hh*:*mm*:*ss*
해당 사항 없음	9 또는 109	*mon dd yyyy hh:mi:ss:mmm*AM(또는 PM)
10	110	*mm-dd-yy*
11	111	*yy*/*mm*/*dd*
12	112	*yymmdd*

디폴트값(*style* 0 또는 100)과 *style* 9 또는 109는 항상 세기를 포함한 값(*yyyy*)을 return합니다. smalldatetime으로부터 char 또는 varchar로 변환하는 경우 초 또는 밀리초 단위를 포함하는 유형은 그 위치에 영(0)을 표시합니다.

예제

예제 1

```
select title, convert(char(12), total_sales)
from titles
```

예제 2

```
select title, total_sales
from titles
where convert(char(20), total_sales) like "1%"
```

예제 3

```
select convert(char(12), getdate(), 3)
```

현재 날짜를 유형 "3", *dd/mm/yy*로 변환합니다.

예제 4

```
select convert(varchar(12), pubdate, 3) from titles
```

pubdate 값이 null이 될 수 있는 경우에는 char보다는 varchar를 사용합니다. 그렇지 않으면 에러가 발생할 수 있습니다.

예제 5

```
select convert(integer, 0x00000100)
```

"0x00000100"문자열에 해당하는 정수를 return합니다. 플랫폼에 따라 다른 결과가 나타날 수 있습니다.

예제 6

```
select convert (binary, 10)
```

Sybase 이진 유형으로서 플랫폼마다 고유한 비트 패턴을 return합니다.

예제 7

```
select convert(bit, $1.11)
```

$1.11에 해당하는 비트 문자열로서 1을 return합니다.

예제 8

```
select title, convert (char(100) not null,
total_sales) into #tempsales
from titles
```

char(100) 데이터 유형의 total_sales와 함께 #tempsales를 만들어내고 null 값을 허용하지 않습니다. titles.total_sales가 null 값을 허용하도록 정의되어 있더라도 #tempsales는 null 값을 허용하지 않는 #tempsales.total_ sales와 함께 생성됩니다.

사용법

- convert는 데이터 유형 변환 함수이며 여러 데이터 유형 간에 변환하고 날짜/시간 및 통화의 표시 형식을 재지정합니다.
- 데이터 유형 변환에 대한 자세한 정보는 51 페이지의 "데이터 유형 변환 함수"를 참조하십시오.
- convert()는 함수가 정의되어 있는 범위 밖에 인자가 있으면 도메인 에러를 생성합니다. 이러한 경우가 발생하지 않도록 해야 합니다.
- null 또는 not null을 사용하여 대상 열의 null 값 허용 여부를 지정하십시오. 특히 이 작업은 select into와 함께 사용하여 원본 테이블에 있는 열의 null 값 허용 여부 및 데이터 유형을 바꾸고 새로운 테이블을 만들 수 있습니다(위의 예제8 참고).
- convert를 사용하여 image 열을 binary나 varbinary로 변환할 수 있습니다. binary 데이터 유형의 최대 길이는 서버의 논리적 페이지 크기에 대한 최대 열 크기로 제한됩니다. 길이를 지정하지 않는 경우 변환된 값의 디폴트 길이는 30자입니다.
- Unichar 표현식은 대상 데이터 유형으로 사용하거나 다른 데이터 유형으로 변환할 수 있습니다. Unichar 표현식은 서버가 지원하는 모든 데이터 유형 간에 명시적 또는 묵시적으로 변환할 수 있습니다.
- unichar이 대상 유형으로 사용되는 경우 길이가 지정되지 않았으면 디폴트 길이인 30 Unicode 값이 사용됩니다. 대상 유형의 길이가 주어진 표현식에 비해 충분하지 않은 경우 에러 메시지가 표시됩니다.

Java 클래스와 관련된 변환

- 데이터베이스에서 Java를 사용할 수 있을 경우 convert는 다음과 같은 방법으로 데이터 유형을 바꿉니다.
  - Java 객체 유형을 SQL 데이터 유형으로 변환
  - SQL 데이터 유형을 Java 유형으로 변환
  - 표현식의 컴파일 시간 데이터 유형(원본 클래스)이 대상 클래스의 하위 클래스나 상위 클래스인 경우 Adaptive Server에 설치되어 있는 Java-SQL 클래스를 Adaptive Server에 설치되어 있는 다른 Java-SQL 클래스로 변환합니다.

  변환 결과는 현재 데이터베이스와 관련되어 있습니다.
- Java 클래스와 관련된 데이터 유형 변환에 대한 정보와 허용되는 데이터 유형 매핑 목록을 보려면 *Java in Adaptive Server Enterprise*를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 convert 함수를 실행할 수 있습니다.

참조 *데이터 유형* – 사용자 정의 데이터 유형

함수 – hextoint, inttohex

COS

설명 지정된 각도의 코사인을 return합니다.

구문 cos(*angle*)

매개변수 *angle*
근사 숫자(float, real 또는 double precision) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

```
select cos(44)

 0.999843
```

사용법

- cos는 수학 함수이며 지정된 코사인 각도(라디안 단위)를 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 cos 함수를 실행할 수 있습니다.

참조 *함수* – acos, degrees, radians, sin

cot

설명 지정된 각도의 코탄젠트를 return합니다.

구문 cot(*angle*)

매개변수 *angle*
근사 숫자(float, real 또는 double precision) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

```
select cot(90)

--------------------
            -0.501203
```

사용법

- cot는 수학 함수이며 지정된 코탄젠트 각도(라디안 단위)를 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 cot 함수를 실행할 수 있습니다.

참조 *함수* – degrees, radians, sin

count

설명 (구별된)null이 아닌 값이나 선택한 행의 개수를 return합니다.

구문 count([all | distinct] *expression*)

매개변수 all
count를 모든 값에 적용합니다. all이 디폴트입니다.

distinct
count가 적용되기 전에 중복된 값을 제거합니다. distinct는 선택 사항입니다.

expression
열 이름, 상수, 함수 그리고 산술 또는 비트 단위 연산자나 서브쿼리로 연결된 열 이름, 상수, 함수의 조합입니다. 집계에서 표현식은 열 이름인 경우가 많습니다. 자세한 내용은 179 페이지의 "표현식"을 참조하십시오.

예제

예제 1

```
select count(distinct city)
from authors
```

저자가 살고 있는 곳의 다른 도시의 수를 구합니다.

예제 2

```
select type
from titles
group by type
having count(*) > 1
```

titles 테이블에 유형을 나열하지만 도서를 하나만 혹은 전혀 포함하지 않는 유형은 삭제됩니다.

사용법

- count는 집계 함수(aggregate function)이며 열에서 null이 아닌 값의 개수를 찾습니다. 집계 함수에 대한 일반적인 내용은 45 페이지의 "집계 함수(aggregate function)"를 참조하십시오.
- distinct가 지정되어 있을 때는 count가 null이 아닌 고유한 값의 개수를 찾습니다. count는 unichar를 포함한 모든 데이터 유형에 사용할 수 있지만 text와 image에는 사용할 수 없습니다. 계산 시 NULL값은 무시됩니다.
- count(*column_name*)는 빈 테이블과 null 값만 갖고 있는 열, null 값만 갖고 있는 그룹 등에 영(0) 값을 return합니다.
- count(*)는 행의 개수를 찾습니다. count(*)는 아무 인자도 취하지 않고 distinct와 함께 사용될 수 없습니다. 모든 행은 null 값의 존재에 관계없이 계산됩니다.
- 테이블이 조인되는 중이면 count(*)를 **select 목록**에 포함하여 조인된 결과에서 행의 개수를 계산하십시오. 조건에 맞는 테이블의 행 개수를 계산하려면 count(*column_name*)를 사용하십시오.
- count()는 서브쿼리에 대한 존재 여부를 확인하는 데 사용할 수 있습니다. 예를 들면 다음과 같습니다.

```
select * from tab where 0 <
    (select count(*) from tab2 where ...)
```

그러나 count()가 모든 일치하는 값을 세므로 exists 또는 in은 결과를 더 빨리 return할 수도 있습니다. 예를 들면 다음과 같습니다.

```
select * from tab where exists
    (select * from tab2 where ...)
```

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 count 함수를 실행할 수 있습니다.

참조 *명령* – compute **절**, group by **및** having **절**, select, where **절**

curunreservedpgs

설명 지정된 디스크에서 사용 가능한 페이지 수를 return합니다.

구문 curunreservedpgs*(dbid*, *lstart, unreservedpgs*)

매개변수

dbid
데이터베이스에 대한 ID입니다. 이들은 sysdatabases의 db_id 열에 저장되어 있습니다.

lstart
페이지가 return될 디스크 내의 페이지입니다.

unreservedpgs
요청된 데이터베이스에 대해 *dbtable*을 사용할 수 없을 경우 return되는 디폴트값입니다.

예제 **예제 1**

```
select db_name(dbid), d.name,
    curunreservedpgs(dbid, lstart, unreservedpgs)
        from sysusages u, sysdevices d
        where d.low <= u.size + vstart
            and d.high >= u.size + vstart -1
            and d.status &2 = 2

master          master       184
master          master       832
tempdb          master       464
tempdb          master      1016
tempdb          master       768
model           master       632
sybsystemprocs  master      1024
pubs2           master       248
```

데이터베이스 이름과 디바이스 이름, 각 디바이스 프래그먼트에 대해 예약되지 않은 페이지 수를 return합니다.

예제 2

```
select curunreservedpgs (dbid, sysusages.lstart, 0)
```

sysusages.lstart에서 시작하는 *dbid*에 대한 세그먼트에 사용할 수 있는 페이지 수를 표시합니다.

사용법

- curunreservedpgs는 시스템 함수이며 디스크에서 사용 가능한 페이지 수를 return합니다. 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.
- 데이터베이스가 열려 있으면 메모리에서 값을 가져옵니다. 데이터베이스가 사용되고 있지 않으면 *sysusages*의 *unreservedpgs* 열에서 값을 가져옵니다.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 curunreservedpgs 함수를 실행할 수 있습니다.

참조 *함수* – db_id, lct_admin

data_pgs

설명 지정된 테이블이나 인덱스에서 사용하는 페이지 수를 return합니다.

구문

data_pgs(*object_id*,
{*data_oam_pg_id* | *index_oam_pg_id*})

매개변수

object_id
테이블, 뷰 또는 기타 데이터베이스 객체에 대한 ID입니다. 객체 ID는 sysobjects의 id 열에 저장됩니다.

data_oam_pg_id
sysindexes의 doampg 열에 저장된 데이터 OAM 페이지에 대한 ID입니다.

index_oam_pg_id
sysindexes의 ioampg 열에 저장된 데이터 OAM 페이지에 대한 ID입니다.

예제

예제 1

```
select sysobjects.name,
Pages = data_pgs(sysindexes.id, doampg)
from sysindexes, sysobjects
where sysindexes.id = sysobjects.id
    and sysindexes.id > 100
    and (indid = 1 or indid = 0)
```

사용자 테이블에서 사용한 데이터 페이지(객체 ID를 100개 넘게 보유)의 수를 추정합니다. 영(0)에 해당하는 indid는 clustered 인덱스가 없는 테이블을 가리키고 1에 해당하는 indid는 clustered 인덱스가 있는 테이블을 가리킵니다. 이 예제에는 nonclustered 인덱스나 텍스트 체인이 없습니다.

예제 2

```
select sysobjects.name,
Pages = data_pgs(sysindexes.id, ioampg)
from sysindexes, sysobjects
where sysindexes.id = sysobjects.id
    and sysindexes.id > 100
    and (indid > 1)
```

사용자 테이블에서 사용한 데이터 페이지(객체 ID를 100개 넘게 보유)와 nonclustered 인덱스, page chain 등의 개수를 추정합니다.

사용법

- data_pgs는 시스템 함수이며 테이블(*doampg*)이나 인덱스(*ioampg*)에서 사용한 페이지 수를 return합니다. sysindexes 테이블에 대해 실행된 쿼리에서는 이 함수를 반드시 사용해야 합니다. 시스템 함수에 대한 자세한 사항은 64 페이지의 "시스템 함수"를 참조하십시오.
- data_pgs는 현재 데이터베이스의 객체에 대해서만 작동됩니다.
- 내부 구조에 대해 사용된 페이지는 결과에 포함되지 않습니다. 테이블과 clustered 인덱스, 내부 구조 등에 대한 페이지 개수 보고서를 보려면 used_pgs를 사용하십시오.

결과의 정확도

- 트랜잭션 로그(syslogs)에서 사용된 경우 결과는 정확하지 않을 수 있으며 16 페이지까지만 사용될 수 있습니다.

에러

- 다음 중 하나가 참일 경우 data_pgs는 에러 대신 영(0)을 return합니다.
  - *object_id*가 sysobjects에 없는 경우
  - *control_page_id*가 *object_id*에서 지정한 테이블에 속하지 않는 경우
  - *object_id*가 -1인 경우
  - *page_id*가 -1인 경우

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 data_pgs 함수를 실행할 수 있습니다.

참조 *함수* – object_id, rowcnt

시스템 프로시저 – sp_spaceused

datalength

설명 지정된 열이나 문자열의 실제 길이를 바이트로 return합니다.

구문 datalength(*expression*)

매개변수 *expression*

열 이름, 변수, 상수 표현식 또는 단일 값으로 계산되는 이들의 조합입니다. 어떤 데이터 유형도 가능합니다. *expression*은 주로 열 이름입니다. *expression*이 문자 상수이면 따옴표로 묶어 주어야 합니다.

예제

```
select Length = datalength(pub_name)
from publishers

Length
-----------
         13
         16
         20
```

publishers 테이블에서 pub_name 열의 길이를 찾습니다.

사용법

- datalength는 시스템 함수이며 *expression*의 길이를 바이트로 return합니다.
- datalength는 각 행에 저장되어 있는 데이터의 실제 길이를 찾습니다. datalength는 varchar, univarchar, varbinary, text, image 등의 데이터 유형에서 유용한데 이는 이들 데이터 유형에서 변수 길이를 저장할 수 있고 후미 공백을 저장하지 않기 때문입니다. char 또는 unichar 값이 null을 허용하도록 선언되면 Adaptive Server는 이를 내부적으로 varchar 또는 univarchar로 저장합니다. 나머지 모든 데이터 유형에 대해서는 datalength가 각각 정의된 길이를 보고합니다.
- NULL 데이터의 datalength는 NULL을 return합니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 datalength 함수를 실행할 수 있습니다.

참조

함수 – char_length, col_length

dateadd

설명

년도와 분기, 시간, 기타 지정된 날짜의 부분 등에 대해 제공된 수를 추가함으로써 생성된 날짜를 return합니다.

구문

dateadd(*date_part*, *integer*, *date*)

매개변수

date_part
날짜 부분이나 약어입니다. Adaptive Server에서 인식한 날짜 부분과 약어의 목록을 보려면 59 페이지의 "날짜 부분"을 참조하십시오.

numeric
정수 표현식입니다.

date
getdate 함수, 사용 가능한 날짜 형식 중 하나의 문자열, 유효한 날짜 형식을 검사하는 표현식 또는 datetime 열의 이름 중 하나입니다.

예제

```
select newpubdate = dateadd(day, 21, pubdate)
from titles
```

titles 테이블에서 모든 도서의 출판일이 21일 경과될 때 새로운 출판일이 표시됩니다.

사용법

- dateadd는 날짜 함수이며 지정된 날짜에 간격을 추가합니다. 날짜 함수에 대한 자세한 내용은 59 페이지의 "날짜 함수"를 참조하십시오.
- dateadd는 날짜 부분과 숫자, 날짜의 세 가지 인자를 취합니다. 결과는 datetime 값으로서 날짜 부분의 수와 날짜를 합한 값에 해당합니다.

  날짜 인자가 smalldatetime 값이면 결과도 smalldatetime이 됩니다. dateadd를 사용하여 smalldatetime에 초나 밀리초 단위를 추가할 수 있으나 1분 이상의 dateadd 변경 사항에 의해 결과 날짜가 return된 경우에만 의미가 있습니다.
- 1753년 1월 1일 이후의 날짜에 대해서만 datetime 데이터 유형을 사용하십시오. datetime 값은 작은 따옴표 또는 큰 따옴표에 포함되어야 합니다. 이전 날짜에는 char, nchar, varchar 또는 nvarchar를 사용합니다. Adaptive Server는 다양한 형식의 날짜를 인식합니다. 자세한 내용은 38 페이지의 "사용자 정의 데이터 유형" 및 51 페이지의 "데이터 유형 변환 함수"를 참조하십시오.

  Adaptive Server는 문자 값을 datetime 값과 비교하려는 경우와 같이 필요한 경우에 문자와 datetime 값을 자동으로 변환합니다.
- weekday 또는 dw와 같은 날짜 부분을 dateadd와 함께 사용하는 것은 논리적이지 않으며 잘못된 결과를 가져옵니다. 대신 day 또는 dd를 사용하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 dateadd 함수를 실행할 수 있습니다.

참조

데이터 유형 – 날짜 및 시간 데이터 유형

명령 – select, where **절**

함수 – datediff, datename, datepart, getdate

datediff

설명

두 날짜 간의 차이를 return합니다.

구문

datediff(*datepart*, *date1*, *date2*)

매개변수

datepart
날짜 부분이나 약어입니다. Adaptive Server에서 인식한 날짜 부분과 약어의 목록을 보려면 59 페이지의 "날짜 부분"을 참조하십시오.

date1
수용 가능한 날짜 형식의 문자열, 유효한 날짜 형식으로 값을 구하는 표현식인 getdate함수나 datetime 열의 이름이 될수 있습니다.

date2
수용 가능한 날짜 형식의 문자열, 유효한 날짜 형식으로 값을 구하는 표현식인 getdate 함수나 datetime 또는 smalldatetime 열의 이름이 될수 있습니다.

예제

```
select newdate = datediff(day, pubdate, getdate())
        from titles
```

이 쿼리는 pubdate와 현재 날짜(getdate 함수로 확인) 간에 경과된 일수를 찾습니다.

사용법

- datediff는 날짜 함수이며 지정된 두 날짜 간에 날짜 부분 수를 계산합니다. 날짜 함수에 대한 자세한 내용은 59 페이지의 "날짜 함수"를 참조하십시오.
- datediff는 세 가지 인자를 취합니다. 첫 번째 인자는 날짜 부분이고 두 번째와 세 번째 인자는 날짜입니다. 결과는 날짜 부분에서 *date2-date1*의 값과 동일한 정수 값으로서 부호가 함께 표시됩니다.
- datediff는 int라는 데이터 유형을 결과로 표시하며 이 결과가 2,147,483,647보다 클 때는 에러가 발생합니다. 밀리초 단위의 경우 대략 24일, 20:31.846시간이 됩니다. 초 단위의 경우는 68년, 19일, 3:14:07시간이 됩니다.
- datediff 결과는 날짜 부분의 배수가 아닌 경우 truncation될 뿐 반올림되지는 않습니다. 예를 들어 hour를 날짜 부분으로 사용하면 "4:00AM"과 "5:50AM"간의 차이는 1이 됩니다.

  날짜 부분으로 day를 사용할 때는 datediff가 지정된 두 시간 사이의 자정 수를 계산합니다. 예를 들어 1992년 1월 1일 23:00시와 1992년 1월 2일 01:00시 사이의 차이는 1이고 1992년 1월 1일 00:00시와 1992년 1월 1일 23:59시 사이의 차이는 영(0)이 됩니다.
- month 날짜 부분은 두 날짜 간의 월의 첫 날 수를 계산합니다. 예를 들어 1월 25일과 2월 2일 사이의 차이는 1이고, 1월 1일과 1월 31일 사이의 차이는 영(0)입니다.

- week 날짜 부분을 datediff와 함께 사용하면 첫 날이 아닌 두 번째 날짜를 포함하여 두 날짜 간에 일요일의 수를 알 수 있습니다. 예를 들어, 1월 4일 일요일과 1월 11일 일요일 사이의 주 수는 1이 됩니다.
- smalldatetime 값이 사용되는 경우 이들은 내부적으로 datetime 값으로 변환되어 계산될 수 있습니다. smalldatetime 값의 초 및 밀리초 단위는 차이 계산에 있어 영(0)으로 자동 설정됩니다.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 datediff 함수를 실행할 수 있습니다.

참조 *데이터 유형* – 날짜 및 시간 데이터 유형

명령 – select, where **절**

함수 – dateadd, datename, datepart, getdate

datename

설명 datetime 값에서 지정된 부분의 이름을 return합니다.

구문 datename (*datepart*, *date*)

매개변수

datepart
날짜 부분이나 약어입니다. Adaptive Server에서 인식한 날짜 부분과 약어의 목록을 보려면 59 페이지의 "날짜 부분"을 참조하십시오.

date
수용 가능한 날짜 형식의 문자열, 유효한 날짜 형식으로 값을 구하는 표현식인 getdate 함수나 datetime 또는 smalldatetime 열의 이름이 될 수 있습니다.

예제

```
select datename(month, getdate())

November
```

이 예제에서는 현재 날짜를 2000년 11월 20일로 가정합니다.

사용법

- datename은 날짜 함수이며 datetime 또는 smalldatetime 값에서 지정된 부분의 이름(예: "6월")을 문자열로 return합니다. 결과로 나타난 날짜가 "23"과 같은 숫자일 때에도 문자열로서 return됩니다.

- 날짜 함수에 대한 자세한 정보는 59 페이지의 "날짜 함수"를 참조하십시오.
- weekday 또는 dw라는 날짜 부분은 datename과 함께 사용될 때 요일(일요일, 월요일 등)을 return합니다.
- smalldatetime이 분 단위까지만 정확하기 때문에 smalldatetime 값과 datename이 함께 사용될 때 초 및 밀리초 단위는 항상 영(0)이 됩니다.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 datename 함수를 실행할 수 있습니다.

참조

데이터 유형 – 날짜 및 시간 데이터 유형

명령 – select, where **절**

함수 – dateadd, datename, datepart, getdate

datepart

설명 datetime 값에서 지정된 부분의 정수 값을 return합니다.

구문 datepart(*date_part*, *date*)

매개변수

date_part
날짜 부분입니다. 표 3-3이 날짜 부분과 datepart에서 인식한 약어, 수용할 수 있는 값 등을 목록으로 나타냅니다.

표 3-3: 날짜 부분과 해당 값

날짜 부분	약어	값
year	yy	1753 – 9999(smalldatetime에서는 2079)
quarter	qq	1 – 4
month	mm	1 – 12
week	wk	1 – 54
day	dd	1 – 31
dayofyear	dy	1 – 366
weekday	dw	1 – 7(일요일-토요일)
hour	hh	0 – 23
minute	mi	0 – 59
second	ss	0 – 59
millisecond	ms	0 – 999
calweekofyear	cwk	1–53
calyearofweek	cyr	1753 – 9999
caldayofweek	cdw	1 – 7

두 자리 숫자(*yy*)로 년도를 입력하는 경우

- 50 미만의 수는 20*yy*로 해석됩니다. 예를 들어 `01`은 2001, `32`는 2032, `49`는 2049를 나타냅니다.
- 50 이상의 숫자는 19*yy*로 해석됩니다. 예를 들어, `50`은 1950, `74`는 1974 그리고 `99`는 1999입니다.

  밀리초 단위는 콜론이나 마침표 다음에 올 수 있습니다. 콜론을 앞에 사용하는 경우 숫자는 천분의 일초를 나타내며 마침표를 앞에 사용하는 경우 한 자리 숫자는 십분의 일초, 두 자리 수는 백분의 일초, 세 자리 숫자는 천분의 일초를 나타냅니다. 예를 들어, "12:30:20:1"은 12:30에서 20과 천분의 1초가 지난 것을 의미하고 "12:30:20.1"은 12:30에서 20과 십분의 1초가 지난 것을 의미합니다.

date

수용 가능한 날짜 형식의 문자열, 유효한 날짜 형식으로 값을 구하는 표현식인 getdate 함수나 datetime 또는 smalldatetime 열의 이름이 될 수 있습니다.

예제

예제 1

```
select datepart(month, getdate())

-----------
         11
```

이 예제에서는 현재 날짜를 1995년 11월 25일로 가정합니다.

예제 2

```
select datepart(year, pubdate) from titles where type =
"trad_cook"

 -----------
        1990
        1985
        1987
```

예제 3

```
select datepart(cwk, '1993/01/01')

-----------
         53
```

예제 4

```
select datepart(cyr,'1993/01/01')

-----------
        1992
```

예제 5

```
select datepart(cdw, '1993/01/01')

-----------
          5
```

사용법

- datepart는 날짜 함수이며 datetime 값에서 지정된 부분에 대한 정수 값을 return합니다. 날짜 함수에 대한 자세한 내용은 59 페이지의 "날짜 함수"를 참조하십시오.
- datepart는 ISO 표준 8601을 따르는 숫자를 return하는데 이 표준은 주의 첫 요일과 년도의 첫 주를 정의합니다. datepart 함수에 calweekofyear, calyearofweek, caldayorweek 값이 포함되는지에 따라 동일한 시간 단위와 return된 날짜가 다를 수 있습니다. 예를 들어, Adaptive Server가 미국 영어를 디폴트 언어로 사용하도록 구성되어 있다고 가정합니다.

  ```
  datepart(cyr, "1/1/1989")
  ```

  위의 경우 1988이 return됩니다. 그러나

  ```
  datepart(yy, "1/1/1989)
  ```

  위의 경우 1989가 return됩니다.

이와 같은 불일치는 ISO 표준에서 목요일을 포함할 *뿐만 아니라* 월요일로 시작하도록 년도의 첫 주를 정의하기 때문에 발생한 것입니다.

디폴트 언어로 미국 영어를 사용하는 서버의 경우 주의 첫 날이 일요일로 시작되도록 하는 년도의 첫 주는 1월 4일이 포함된 주입니다.

- weekday 또는 dw의 날짜 부분은 datepart와 함께 사용할 때 해당 숫자를 return합니다. 요일의 이름에 해당하는 숫자는 datefirst 설정에 따라 달라집니다. 일부 언어 디폴트(us_english 포함)는 일요일=1, 월요일=2와 같은 식으로 나타내고, 그 외 다른 언어는 월요일=1, 화요일=2와 같은 식으로 나타냅니다. 디폴트 동작은 set datefirst를 사용한 세션별 기준에 따라 달라질 수 있습니다.
- calweekofyear(cwk로 축약될 수 있음)는 한 년도 내 주의 순서상 위치를 return합니다. calyearofweek(cyr로 축약될 수 있음)는 주가 시작되는 년도를 return합니다. caldayofweek(cdw로 축약될 수 있음)는 주 내 요일의 순서상 위치를 return합니다. calweekofyear, calyearofweek, caldayofweek는 dateadd, datediff, datename에 대한 날짜 부분으로 사용할 수 없습니다.
- smalldatetime이 분 단위까지만 정확하기 때문에 smalldatetime 값과 datepart가 함께 사용될 때 초 및 밀리초 단위는 항상 영(0)이 됩니다.
- 요일의 날짜 부분 값은 언어 설정에 따라 다릅니다.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 datepart 함수를 실행할 수 있습니다.

참조 *데이터 유형* – 날짜 및 시간 데이터 유형

명령 – select, where **절**

함수 – dateadd, datediff, datename, getdate

db_id

설명 지정된 데이터베이스의 ID 번호를 return합니다.

구문 db_id(*database_name*)

매개변수

database_name
데이터베이스의 이름입니다. *database_name*은 문자 표현식이어야 합니다. 상수 표현식일 경우에는 따옴표를 사용해야 합니다.

예제

```
select db_id("sybsystemprocs")

------
4
```

사용법

- db_id는 시스템 함수이며 데이터베이스 ID 번호를 return합니다.
- *database_name*을 따로 지정하지 않은 경우에는 db_id가 현재 데이터베이스의 ID 번호를 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 db_id 함수를 실행할 수 있습니다.

참조

함수 – db_name, object_id

db_name

설명

ID 번호가 지정되어 있는 데이터베이스의 이름을 return합니다.

구문

db_name([*database_id*])

매개변수

database_id
데이터베이스 ID(sysdatabases.dbid에 저장되어 있음)에 대한 숫자 표현식입니다.

예제

예제 1

```
select db_name()
```

현재 데이터베이스의 이름을 return합니다.

예제 2

```
select db_name(4)


------------------------------
sybsystemprocs
```

사용법

- db_name은 시스템 함수이며 데이터베이스 이름을 return합니다.

- *database_id*를 입력하지 않은 경우에는 db_name이 현재 데이터베이스의 이름을 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 db_name 함수를 실행할 수 있습니다.

참조 *함수* – col_name, db_id, object_name

degrees

설명 지정된 라디안 단위로 각도의 크기를 도(degree)로 return합니다.

구문 degrees(*numeric*)

매개변수 *numeric*
도(degree)로 변환될 라디안 단위의 숫자입니다.

예제

```
select degrees(45)

-----------
       2578
```

사용법

- degrees는 수학 함수이며 라디안을 도(degree)로 변환합니다. 결과 값은 숫자 표현식과 같은 유형입니다.

  숫자 및 Decimal 표현식의 경우 결과에 77이라는 내부 정밀도(P)가 나타나고 소수점 이하 자릿수는 표현식의 소수점 이하 자릿수와 같게 됩니다.

  통화 데이터 유형이 사용될 경우 float 유형으로 내부 변환을 수행하면 정밀도(P)를 잃을 수도 있습니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 degrees 함수를 실행할 수 있습니다.

참조 *함수* – radians

difference

설명

두 soundex 값 사이의 차이를 return합니다.

구문

difference(*char_expr1*|u*char_expr1*), (*char_expr2*| *uchar_expr2*)

매개변수

char_expr1
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

char_expr2
다른 문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형의 상수 표현식입니다.

uchar_expr1
문자 유형 열 이름, 변수 또는 unichar 유형 등의 상수 표현식입니다.

uchar_expr2
다른 문자 유형 열 이름, 변수 또는 unichar 유형 등의 상수 표현식입니다.

예제

예제 1

```
select difference("smithers", "smothers")

---------
4
```

예제 2

```
select difference("smothers", "brothers")

---------
2
```

사용법

- difference는 문자열 함수이며 두 soundex 값 사이의 차이를 나타내는 정수를 return합니다.
- difference 함수는 두 문자열을 비교하여 유사성을 평가한 다음 0부터 4까지의 값을 return합니다. 가장 일치하는 경우는 4가 됩니다.

  문자열 값은 유효한 단일 바이트 또는 더블 바이트의 로마 문자로 구성되어야 합니다.
- *char_expr1, uchar_expr1* 또는 *char_expr2, uchar_expr2*가 NULL이면 NULL이 return됩니다.

- varchar 표현식이 하나의 매개변수로 주어지고 unichar 표현식이 다른 매개변수로 주어진 경우 varchar 표현식은 묵시적으로 unichar 표현식으로 변환되며 일부 truncation될 수 있습니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 difference 함수를 실행할 수 있습니다.

참조 *함수* – soundex

함수: *exp – mut_excl_roles*

exp

설명

상수 e를 지정된 수로 제곱한 값을 return합니다.

구문

exp(*approx_numeric*)

매개변수

approx_numeric

근사 숫자(float, real 또는 double precision) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

```
select exp(3)

--------------------
             20.085537
```

사용법

- exp는 수학 함수이며 지정된 값의 지수 값을 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 exp 함수를 실행할 수 있습니다.

참조

함수 – log, log10, power

floor

설명

지정된 값보다 작거나 같은 값 중 가장 큰 정수를 return합니다.

구문

floor(*numeric*)

매개변수

numeric

정밀 숫자(numeric, dec, decimal, tinyint, smallint 또는 int), 근사 숫자(float, real 또는 double precision) 또는 money 유형의 열, 변수, 상수 표현식 또는 이들의 조합입니다.

예제

예제 1

```
select floor(123)

-----------
        123
```

예제 2

```
select floor(123.45)

-------
    123
```

예제 3

```
select floor(1.2345E2)

--------------------
         123.000000
```

예제 4

```
select floor(-123.45)

-------
   -124
```

예제 5

```
select floor(-1.2345E2)

--------------------
        -124.000000
```

예제 6

```
select floor($123.45)

------------------------
                123.00
```

사용법

- floor는 수학 함수이며 지정된 값보다 작거나 같은 값 중 가장 큰 정수를 return합니다. 결과 값은 숫자 표현식과 같은 유형입니다.

  숫자 표현식과 Decimal 표현식의 경우 결과 값은 표현식과 같은 정밀도(P)와 0의 소수점 이하 자릿수를 갖습니다.

- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 floor 함수를 실행할 수 있습니다.

참조 *함수* – abs, ceiling, round, sign

getdate

설명 현재 시스템 날짜 및 시간을 return합니다.

구문 getdate( )

매개변수 없음.

예제

예제 1

```
select getdate()
Nov 25 1995 10:32AM
```

예제 2

```
select datepart(month, getdate())
1
```

예제 3

```
select datename(month, getdate())
November
```

이 예제는 현재 날짜를 1995년 11월 25일 오전 10시 32분으로 가정합니다.

사용법

- getdate는 날짜 함수이며 현재 시스템 날짜 및 시간을 return합니다.
- 날짜 함수에 대한 자세한 정보는 59 페이지의 "날짜 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 getdate 함수를 실행할 수 있습니다.

참조 *데이터 유형* – 날짜 및 시간 데이터 유형

함수 – dateadd, datediff, datename, datepart

hextoint

설명 16진수 문자열에 해당하는 플랫폼 독립적인 정수를 return합니다.

구문 hextoint (*hexadecimal_string*)

매개변수 *hexadecimal_string*
정수로 변환할 16진수 값입니다. 문자 유형 열, 변수 이름 또는 인용 부호로 묶인 유효한 16진수 문자열("0x" 접두어가 있거나 없음)이어야 합니다.

예제

```
select hextoint ("0x00000100")
```

16진수 문자열 "0x00000100"에 해당하는 정수를 return합니다. 결과 값은 실행된 플랫폼에 관계없이 항상 256입니다.

사용법

- hextoint는 데이터 유형 변환 함수이며 16진수 문자열에 해당하는 플랫폼 독립적인 정수를 return합니다.
- 16진수 데이터를 플랫폼 독립적인 정수로 변환하려면 hextoint 함수를 사용하십시오. hextoint 함수는 인용 부호로 묶인 유효한 16진수 문자열("0x" 접두어가 있거나 없음) 또는 문자 유형 열이나 변수 이름을 사용합니다.

  hextoint 함수는 16진수 문자열에 해당하는 정수를 return합니다. 이 함수는 항상 실행된 플랫폼에 관계없이 주어진 16진수 문자열에 해당하는 정수를 return합니다.
- 데이터 유형 변환에 대한 자세한 내용은 51 페이지의 "데이터 유형 변환 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 hextoint 함수를 실행할 수 있습니다.

참조 *함수* – convert, inttohex

host_id

설명 호스트 프로세스 ID 또는 클라이언트 프로세스를 return합니다.

구문 host_id()

매개변수 없음.

예제

```
select host_id()

------------------------------
24711
```

사용법

- host_id는 시스템 함수이며 클라이언트 프로세스(서버 프로세스가 아님)의 호스트 프로세스 ID를 return합니다.
- 시스템 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 host_id 함수를 실행할 수 있습니다.

참조

함수 – host_name

host_name

설명

클라이언트 프로세스의 현재 호스트 컴퓨터 이름을 return합니다.

구문

host_name( )

매개변수

없음.

예제

```
select host_name()

------------------------------
violet
```

사용법

- host_name은 시스템 함수이며 클라이언트 프로세스(서버 프로세스가 아님)의 현재 호스트 컴퓨터 이름을 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 host_name 함수를 실행할 수 있습니다.

참조

함수 – host_id

index_col

설명

지정된 테이블이나 뷰에 있는 인덱스된 열의 이름을 return합니다.

구문

index_col (*object_name*, *index_id*, *key_#* [, *user_id*])

매개변수

object_name
테이블이나 뷰의 이름입니다. 완전히 한정된 이름(즉, 데이터베이스와 소유자 이름을 포함할 수 있음)으로 지정할 수 있습니다. 이때 이름은 인용 부호로 묶어야 합니다.

index_id
object_name 인덱스의 번호입니다. 이 번호는 sysindexes.indid 값과 같습니다.

key_#
인덱스의 키입니다. 이 값은 clustered 인덱스에 대해서는 1과 sysindexes.keycnt 사이에 있으며, nonclustered 인덱스에 대해서는 1과 sysindexes.keycnt+1 사이에 있습니다.

user_id
*object_name*의 소유자입니다. *user_id*를 지정하지 않으면 호출자의 사용자 ID가 디폴트값으로 사용됩니다.

예제

```
declare @keycnt integer
select @keycnt = keycnt from sysindexes
     where id = object_id("t4")
     and indid = 1
while @keycnt > 0
begin
     select index_col("t4", 1, @keycnt)
     select @keycnt = @keycnt - 1
 end
```

테이블 t4의 clustered 인덱스에 있는 키 이름을 찾습니다.

사용법

- index_col은 시스템 함수이며 인덱스된 열 이름을 return합니다.
- *object_name*이 테이블이나 뷰 이름이 아니면 index_col에서 NULL이 return됩니다.
- 시스템 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 index_col 함수를 실행할 수 있습니다.

참조

함수 – object_id

시스템 프로시저 – sp_helpindex

index_colorder

설명

열 순서를 return합니다.

구문

index_colorder (*object_name*, *index_id*, *key_#*
[, *user_id*])

매개변수

object_name
테이블이나 뷰의 이름입니다. 완전히 한정된 이름(즉, 데이터베이스와 소유자 이름을 포함할 수 있음)으로 지정할 수 있습니다. 이때 이름은 인용 부호로 묶어야 합니다.

index_id
object_name 인덱스의 번호입니다. 이 번호는 sysindexes.indid 값과 같습니다.

key_#
인덱스의 키입니다. 1과 인덱스의 키 개수를 유효한 값으로 사용할 수 있습니다. 키 개수는 sysindexes.keycnt에 저장됩니다.

user_id
*object_name*의 소유자입니다. *user_id*를 지정하지 않으면 호출자의 사용자 ID가 디폴트값으로 사용됩니다.

예제

```
select name, index_colorder("sales", indid, 2)
from sysindexes
where id = object_id ("sales")
and indid > 0

name
------------------------- -------------------------
salesind                  DESC
```

sales 테이블의 salesind 인덱스가 내림차순이므로 "DESC"를 return합니다.

사용법

- index_colorder 시스템 함수는 오름차순 열에 대해서는 "ASC"를 return하고 내림차순 열에 대해서는 "DESC"를 return합니다.
- *object_name*이 테이블 이름이 아니거나 *key_#*가 유효한 키 번호가 아니면 index_colorder에서 NULL이 return됩니다.
- 시스템 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 index_colorder 함수를 실행할 수 있습니다.

inttohex

설명 지정된 정수에 해당하는 플랫폼 독립적인 16진수를 return합니다.

구문 inttohex (*integer_expression*)

매개변수 *integer_expression*
16진수 문자열로 변환할 정수 값입니다.

예제

```
select inttohex (10)

--------
0000000A
```

사용법

- inttohex는 데이터 유형 변환 함수이며 정수에 해당하는 플랫폼 독립적인 16진수를 "0x" 접두어 없이 return합니다.
- 정수를 플랫폼 독립적인 16진수 문자열로 변환하려면 inttohex 함수를 사용하십시오. 정수로 계산되는 표현식이면 inttohex 함수를 사용할 수 있습니다. 항상 실행된 플랫폼에 관계없이 주어진 표현식에 해당하는 16진수를 return합니다.
- 데이터 유형 변환에 대한 자세한 내용은 51 페이지의 "데이터 유형 변환 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 inttohex 함수를 실행할 수 있습니다.

참조 *함수* – convert, hextoint

isnull

설명 *expression1*이 NULL로 계산되는 경우 *expression2* 에 지정된 값을 대체합니다.

구문 isnull(*expression1*, *expression2*)

매개변수 *expression*
열 이름, 변수, 상수 표현식 또는 단일 값으로 계산되는 이들의 조합입니다. unichar를 포함한 모든 데이터 유형이 될 수 있습니다. *expression*은 일반적으로 열 이름입니다. *expression*이 문자 상수이면 따옴표로 묶어 주어야 합니다.

예제

```
select isnull(price,0)
from titles
```

price에 있는 NULL 값을 0으로 바꾸면서 titles 테이블의 모든 행을 return합니다.

사용법

- isnull은 시스템 함수이며 *expression1*이 NULL로 계산되면 *expression2*에 지정된 값으로 대체합니다. 시스템 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.
- 표현식의 데이터 유형은 묵시적으로 변환하거나 convert 함수를 사용해야 합니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 isnull 함수를 실행할 수 있습니다.

참조

함수 – convert

is_sec_service_on

설명

보안 서비스를 사용하면 1, 사용하지 않으면 0을 return합니다.

구문

is_sec_service_on(*security_service_nm*)

매개변수

security_service_nm
보안 서비스의 이름입니다.

예제

```
select is_sec_service_on("unifiedlogin")
```

사용법

- is_sec_service_on을 사용하여 해당 보안 서비스가 세션 동안 사용되는지 확인합니다.
- 보안 서비스의 유효한 이름을 찾으려면 다음 쿼리를 실행하십시오.

  ```
  select * from syssecmechs
  ```

  결과는 다음과 같이 표시됩니다.

  ```
  sec_mech_name available_service
  ------------- --------------------
  dce           unifiedlogin
  dce           mutualauth
  dce           delegation
  dce           integrity
  dce           confidentiality
  dce           detectreplay
  dce           detectseq
  ```

available_service 열은 Adaptive Server가 지원하는 보안 서비스를 표시합니다.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 is_sec_service_on 함수를 실행할 수 있습니다.

참조 *함수* – show_sec_services

lct_admin

설명 마지막 임계값을 관리합니다.

마지막 임계값의 현재 값을 return합니다.

마지막 임계값에 도달한 트랜잭션 로그의 트랜잭션을 중지합니다.

구문

```
lct_admin({{"lastchance" | "logfull" }, database_id
   | "reserve", {log_pages | 0 }
   | "abort", process-id [, database-id]})
```

매개변수

lastchance
지정된 데이터베이스에 마지막 임계값을 생성합니다.

logfull
마지막 임계값이 지정된 데이터베이스에서 교차된 경우 1을, 아닌 경우 0을 return합니다.

database_id
데이터베이스를 지정합니다.

reserve
마지막 임계값의 현재 값이나 지정된 크기의 트랜잭션 로그를 덤프할 때 필요한 로그 페이지 수를 확보합니다.

log_pages
마지막 임계값을 결정하는 페이지 수입니다.

0
마지막 임계값의 현재 값을 return합니다. 로그와 데이터 세그먼트가 분리된 데이터베이스에서 마지막 임계값의 크기는 동적으로 변하지 않습니다. 이 값은 트랜잭션 로그의 크기를 기준으로 한 고정 값입니다. 로그와 데이터 세그먼트가 혼합된 데이터베이스에서 마지막 임계값은 동적으로 변합니다.

abort
트랜잭션 로그가 마지막 임계값에 도달한 데이터베이스의 트랜잭션을 중지합니다. LOG SUSPEND 모드에 있는 트랜잭션만 중지될 수 있습니다.

process-id
로그 일시 중단 모드의 프로세스 ID(*spid*) 입니다. 프로세스에 LCT(마지막 임계값)에 도달한 트랜잭션 로그에 진행 중인 트랜잭션이 있으면 이 프로세스는 로그 일시 중단 모드로 됩니다.

database-id
트랜잭션 로그가 마지막 임계값에 도달한 데이터베이스의 ID입니다. *process-id*가 0이면 지정된 데이터베이스에서 진행 중인 트랜잭션이 모두 종료됩니다.

예제

예제 1

```
select lct_admin("lastchance", 1)
```

이 예제는 dbid가 1인 데이터베이스에 대한 로그 세그먼트의 마지막 임계값을 생성하고 새 임계값이 설정된 페이지 수를 return합니다. 이전의 마지막 임계값이 있을 경우 새 값으로 바뀝니다.

예제 2

```
select lct_admin("logfull", 6)
```

db_id가 6인 데이터베이스의 마지막 임계값이 초과된 경우 1을, 아닌 경우 0을 return합니다.

예제 3

```
select lct_admin("reserve", 64)

-----------
         16
```

64페이지가 들어 있는 로그에 트랜잭션 로그를 덤프하는 데 필요한 로그 페이지 수를 계산하고 return합니다.

예제 4

```
select lct_admin("reserve",0)
```

명령을 보냈던 데이터베이스에 있는 트랜잭션 로그의 현재 마지막 임계값을 return합니다.

예제 5

```
select lct_admin("abort", 83)
```

프로세스 83의 트랜잭션을 중지합니다. 이 프로세스는 로그 일시 중단 모드에 있어야 합니다. 마지막 임계값에 도달한 트랜잭션 로그의 트랜잭션만 종료됩니다.

예제 6

```
select lct_admin("abort", 0, 5)
```

데이터베이스 ID가 5인 데이터베이스에 열려 있는 트랜잭션을 모두 중지합니다.

이 형식은 로그 세그먼트의 마지막 임계값에서 일시 중단될 수 있는 프로세스를 환기시킵니다.

사용법

- lct_admin은 시스템 함수이며 로그 세그먼트의 마지막 임계값을 관리합니다. 시스템 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.
- lct_admin("lastchance", *dbid*)이 0을 return하면 로그는 이 데이터베이스에서 별도의 세그먼트에 있는 것이 아니므로 마지막 임계값이 존재하지 않습니다.
- 별도의 로그 세그먼트를 가진 데이터베이스를 생성할 때마다 서버에서는 sp_thresholdaction을 디폴트로 호출하는 디폴트 마지막 임계값을 생성합니다. sp_thresholdaction을 호출한 프로시저가 서버에 없을 때에도 디폴트 마지막 임계값이 생성됩니다.

  로그가 마지막 임계값을 초과하는 경우 Adaptive Server에서는 활동을 일시 중단하고, sp_thresholdaction 호출을 시도하여 발견되지 않으면 에러를 생성한 다음 로그를 truncation할 때까지 프로세스를 일시 중단합니다.
- LCT(마지막 임계값)에 도달한 트랜잭션 로그의 열린 트랜잭션 중 가장 오래된 트랜잭션을 종료하려면 이 트랜잭션을 초기화한 프로세스 ID를 입력합니다.
- 마지막 임계값에 도달한 트랜잭션 로그의 열린 트랜잭션을 모두 종료하려면 *process_id*에 0을 입력하고 *database-id* 매개변수에 데이터베이스 ID를 지정합니다.
- 자세한 내용은 시스템 관리 지침서를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한 시스템 관리자만 lct_admin abort 함수를 실행할 수 있습니다. 모든 사용자는 다른 lct_admin 옵션을 실행할 수 있습니다.

참조 *명령* – dump transaction

함수 – curunreservedpgs

license_enabled

설명 함수의 라이센스가 설정되어 있으면 1을, 설정되어 있지 않으면 0을, 유효하지 않은 라이센스 이름을 지정하면 null을 return합니다.

구문 license_enabled("ase_server" | "ase_ha" | "ase_dtm" | "ase_java" | "ase_asm")

매개변수

ase_server
Adaptive Server의 라이센스를 지정합니다.

ase_ha
Adaptive Server 고가용성(HA) 기능의 라이센스를 지정합니다.

ase_dtm
Adaptive Server 분산 트랜잭션 관리 기능의 라이센스를 지정합니다.

ase_java
Adaptive Server Java 기능의 라이센스를 지정합니다.

ase_asm
Adaptive Server 고급 보안 메커니즘의 라이센스를 지정합니다.

예제

```
select license_enabled("ase_dtm")

--------------
           1
```

Adaptive Server 분산 트랜잭션 관리 기능의 라이센스가 사용 가능 상태임을 나타냅니다.

사용법

- Adaptive Server 기능에 대한 라이센스 키 설치에 대한 정보는 *설치 안내서*를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자는 license_name을 실행할 수 있습니다.

참조 *시스템 프로시저* – sp_configure

lockscheme

설명 지정된 객체의 locking 스키마를 문자열로 return합니다.

구문 lockscheme(*object_name*)
또는
lockscheme(*dbid*, *object_id*)

매개변수

oject_name
찾는 객체의 이름입니다. 완전히 유효한 객체 이름을 지정하지 않으면 현재 데이터베이스가 검색됩니다.

dbid
*object_name*에 의해 지정된 데이터베이스의 ID입니다.

object_id
*object_name*에 의해 표시된 객체의 ID입니다.

예제

예제 1

```
select lockscheme(title)
from titles
```

titles 테이블의 title 열의 locking 스키마를 선택합니다.

예제 2

```
select lockscheme(4, 224000798)
```

데이터베이스 ID 4(pubs2 데이터베이스)에서 *object_id* 224000798(이 경우엔 titles 테이블)의 locking 스키마를 선택합니다.

사용법

- lockscheme()는 varchar(11)을 return하고 NULL을 허용합니다.
- 지정된 객체가 테이블이 아닌 경우 lockscheme()는 "not a table" 문자열을 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자는 lockscheme()를 실행할 수 있습니다.

log

설명 지정된 수의 자연 로그를 return합니다.

구문 log(*approx_numeric*)

매개변수

approx_numeric
근사 숫자(float, real 또는 double precision) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

```
select log(20)

--------------------
              2.995732
```

사용법

- log는 수학 함수이며 지정된 값의 자연 로그를 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 log 함수를 실행할 수 있습니다.

참조

함수 – log10, power

log10

설명

지정된 수의 베이스 10 로그를 return합니다.

구문

log10(*approx_numeric*)

매개변수

approx_numeric
근사 숫자(float, real 또는 double precision) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

```
select log10(20)

--------------------
             1.301030
```

사용법

- log10은 수학 함수이며 지정된 값의 베이스 10 로그를 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 log10 함수를 실행할 수 있습니다.

참조

함수 – log, power

lower

설명 지정된 표현식의 소문자 형태를 return합니다.

구문 lower(*char_expr* | *uchar_expr*)

매개변수

char_expr
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

uchar_expr
문자 유형 열 이름, 변수 또는 unichar 또는 univarchar 유형 등의 상수 표현식입니다

예제

```
select lower(city) from publishers

--------------------
boston
Washington
berkeley
```

사용법

- lower는 문자열 함수이며 대문자를 소문자로 변환하여 문자 값을 return합니다.
- lower 함수는 upper 함수와 정반대의 기능을 갖습니다.
- *char_expr* 또는 *uchar_expr*이 NULL이면 NULL을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 lower 함수를 실행할 수 있습니다.

참조 *함수* – upper

ltrim

설명 선두 공백을 잘라내고 지정된 표현식을 return합니다.

구문 ltrim(char_expr|uchar_expr)

매개변수

char_expr
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형 등의 상수 표현식입니다.

예제

```
select ltrim("    123")

-------
123
```

사용법

- ltrim은 문자열 함수이며 문자식에서 선두 공백을 제거합니다. 현재 문자 집합에서 공백 문자에 해당하는 값만 제거합니다.
- *char_expr* 또는 *uchar_expr*이 NULL이면 NULL이 return됩니다.
- Unicode 표현식의 경우 지정된 표현식에 해당하는 소문자 Unicode를 return합니다. 해당 소문자가 없는 표현식의 문자는 수정되지 않습니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 ltrim 함수를 실행할 수 있습니다.

참조

함수 – rtrim

max

설명

표현식에서 가장 큰 값을 return합니다.

구문

max(*expression*)

매개변수

expression
열 이름, 상수, 함수 그리고 산술 또는 비트 단위 연산자나 서브쿼리로 연결된 열 이름, 상수, 함수의 모든 조합입니다.

예제

예제 1

```
select max(discount) from salesdetail

--------------------
            62.200000
```

salesdetail 테이블의 discount 열에서 최대값을 새 열로 return합니다.

예제 2

```
select discount from salesdetail
compute max(discount)
```

salesdetail 테이블의 discount 열에서 최대값을 새 행으로 return합니다.

사용법

- max는 집계 함수(aggregate function)이며 열 또는 표현식에서 최대값을 찾습니다. 집계 함수에 대한 일반적인 내용은 45 페이지의 "집계 함수(aggregate function)"를 참조하십시오.
- max는 정밀 숫자, 근사 숫자, 문자 및 datetime 열이 될 수 있습니다. bit 열과 함께 사용할 수 없습니다. 문자 열을 사용하면 max는 조합 순서에서 가장 높은 값을 찾습니다. max는 null 값은 무시합니다. max는 모든 후미 공백을 잘라내고 char 데이터 유형을 varchar로, unichar 데이터 유형을 univarchar로 묵시적으로 변환합니다.
- unichar 데이터는 디폴트 Unicode 정렬 순서에 따라 조회됩니다.
- Adaptive Server는 다음 경우가 아니면 집계 열에 인덱스가 있을 때 인덱스 끝으로 직접 이동하여 max의 마지막 행을 찾습니다.
  - *expression*이 열이 아닌 경우
  - 열이 인덱스의 첫 번째 열이 아닌 경우
  - 쿼리에 다른 집계가 있는 경우
  - group by 또는 where 절이 있는 경우

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 max 함수를 실행할 수 있습니다.

참조 *명령* – compute **절**, group by **및** having **절**, select, where **절**

함수 – avg, min

min

설명 열에서 가장 낮은 값을 return합니다.

구문 min(*expression*)

매개변수

expression
열 이름, 상수, 함수 그리고 산술 또는 비트 단위 연산자나 서브쿼리로 연결된 열 이름, 상수, 함수의 모든 조합입니다. 집계 함수(aggregate function)와 함께 쓸 때 표현식은 항상 열 이름입니다. 자세한 내용은 179 페이지의 "표현식"을 참조하십시오.

예제

```
select min(price) from titles
where type = "psychology"

------------------------
                    7.00
```

사용법

- min은 집계 함수(aggregate function)이며 열에서 최소값을 찾습니다.
- 집계 함수(aggregate functions)에 대한 일반적인 사항은 45 페이지의 "집계 함수(aggregate function)"를 참조하십시오.
- min은 숫자, 문자 및 datetime 열이 될 수 있습니다. bit 열과 함께 사용할 수 없습니다. 문자열을 사용하면 min은 정렬 순서에서 가장 낮은 값을 찾습니다. min 은 char 데이터 유형을 varchar 로, unichar 데이터 유형을 univarchar로 묵시적으로 변환하며 모든 후미 공백을 잘라냅니다. min은 null 값을 무시합니다. distinct는 min에 대해 의미가 없으므로 사용할 수 없습니다.
- unichar 데이터는 디폴트 Unicode 정렬 순서에 따라 조회됩니다.
- Adaptive Server는 다음 경우가 아니면 집계 열에 인덱스가 있을 때 min의 첫 번째 정규 행으로 직접 이동합니다.
  - *expression*이 열이 아닌 경우
  - 열이 인덱스의 첫 번째 열이 아닌 경우
  - 쿼리에 다른 집계가 있는 경우
  - group by 절이 있는 경우

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 min을 사용할 수 있습니다.

참조

명령 – compute **절**, group by **및** having **절**, select, where **절**

함수 – avg, max

mut_excl_roles

설명

두 롤(role) 사이의 상호 배타성에 대한 정보를 return합니다.

구문

mut_excl_roles (*role1, role2* [membership | activation])

매개변수

role1
상호 배타적 관계에서 사용자 정의된 한 롤(role)입니다.

role2
상호 배타적 관계에서 사용자 정의된 다른 롤(role)입니다.

level
지정된 롤(role)이 배타적인 수준(구성원 또는 활성)입니다.

예제

```
alter role admin add exclusive membership supervisor
select
mut_excl_roles("admin", "supervisor", "membership")

-----------
          1
```

admin과 supervisor 롤(role)이 상호 배타적임을 보여 줍니다.

사용법

- mut_excl_roles는 시스템 함수이며 두 롤(role) 사이의 상호 배타성에 대한 정보를 return 합니다. 시스템 보안 담당자가 role1을 role2 와 상호 배타적으로 정의하거나 role2에 직접 포함된 롤(role)로 정의하는 경우 mut_excl_roles는 1을 return하고, 롤(role)이 상호 배타적이지 않으면 mut_excl_roles가 0을 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자는 mut_excl_roles 함수를 실행할 수 있습니다.

참조

명령 – alter role, create role, drop role, grant, set, revoke

함수 – proc_role, role_contain, role_id, role_name

시스템 프로시저 – sp_activeroles, sp_displayroles, sp_role

함수: *object_id – rtrim*

object_id

설명

지정된 객체의 객체 ID를 return합니다.

구문

object_id(*object_name*)

매개변수

object_name

테이블, 뷰, 프로시저, 트리거, 디폴트 또는 규칙과 같은 데이터베이스 객체의 이름입니다. 완전히 한정된 이름(즉, 데이터베이스와 소유자 이름을 포함할 수 있음)으로 지정할 수 있습니다. *object_ name*을 따옴표로 묶습니다.

예제

예제 1

```
select object_id("titles")

-----------
  208003772
```

예제 2

```
select object_id("master..sysobjects")

-----------
          1
```

사용법

- object_id는 시스템 함수이며 객체의 ID를 return합니다. 객체 ID는 sysobjects의 id 열에 저장됩니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 object_id 함수를 실행할 수 있습니다.

참조

함수 – col_name, db_id, object_name

시스템 프로시저 – sp_help

object_name

설명

객체 ID가 지정된 객체의 이름을 return합니다.

구문

object_name(*object_id*[, *database_id*])

매개변수

object_id

테이블, 뷰, 프로시저, 트리거, 디폴트 또는 규칙과 같은 데이터베이스 객체의 객체 ID입니다. 객체 ID는 sysobjects의 id 열에 저장됩니다.

database_id

객체가 현재 데이터베이스에 없는 경우 데이터베이스에 대한 ID입니다. 데이터베이스 ID는 sysdatabases의 db_id 열에 저장됩니다 .

예제

예제 1

```
select object_name(208003772)

------------------------------
titles
```

예제 2

```
select object_name(1, 1)

------------------------------
sysobjects
```

사용법

- object_name은 시스템 함수이며 객체의 이름을 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 object_name 함수를 실행할 수 있습니다.

참조

함수 – col_name, db_id, object_id

시스템 프로시저 – sp_help

patindex

설명

지정된 패턴이 처음으로 발생하는 시작 위치를 return합니다.

구문

patindex("%*pattern*%", *char_expr* | *uchar_expr* [, using {bytes | characters | chars}])

매개변수

pattern
Adaptive Server에서 지원하는 패턴 일치 대표 문자를 포함하는 char 또는 varchar 데이터 유형의 문자 표현식입니다. 첫 번째 또는 마지막 문자를 검색하는 경우가 아니라면 % 대표 문자가 *pattern*의 앞과 뒤에 와야 합니다. *pattern*에 사용할 수 있는 대표 문자에 대한 설명은 195 페이지의 "대표 문자를 사용한 패턴 일치"를 참조하십시오.

char_expr
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형 등의 상수 표현식입니다.

using
시작 위치에 대한 형식을 지정합니다.

bytes
오프셋을 바이트로 return합니다.

chars 또는 characters – 오프셋을 문자로 return합니다(디폴트).

예제

예제 1

```
select au_id, patindex("%circus%", copy)
from blurbs
```

```
au_id
 ----------- -----------
 486-29-1786           0
 648-92-1872           0
 998-72-3567          38
 899-46-2035          31
 672-71-3249           0
 409-56-7008           0
```

copy 열에 있는 단어 "circus"의 시작 문자 위치와 author ID를 표시합니다.

예제 2

```
select au_id, patindex("%circus%", copy,
     using chars)
from blurbs
```

예제 3

```
select au_id, patindex("%circus%", copy,
     using chars)
from blurbs
```

예제 1과 같습니다.

예제 4

```
select name
from sysobjects
where patindex("sys[a-d]%", name) > 0
```

```
name
------------------------------
sysalternates
sysattributes
syscharsets
syscolumns
syscomments
sysconfigures
sysconstraints
syscurconfigs
sysdatabases
sysdepends
sysdevices
```

sysobjects에서 "sys"로 시작하고 4번째 문자가 "a", "b", "c" 또는 "d"인 모든 행을 찾습니다.

사용법

- patindex는 문자열 함수이며 지정된 문자 표현식에서 *pattern*이 처음으로 나타나는 시작 위치를 표시하는 정수를 return하거나 *pattern*을 찾을 수 없는 경우 0을 return합니다.
- patindex는 text 및 image 데이터를 포함한 모든 문자 데이터에 대해 사용될 수 있습니다.
- 기본적으로 patindex는 오프셋을 문자로 return합니다. 오프셋을 바이트(다중 바이트 문자열)로 return하도록 하려면 using bytes를 지정합니다.
- *pattern*의 앞뒤에 퍼센트 기호를 포함합니다. 열의 첫 번째 문자인 *pattern*을 찾으려면 앞에 오는 %를 생략합니다. 열의 마지막 문자인 *pattern*을 찾으려면 뒤에 오는 %를 생략합니다.
- *char_expr* 또는 *uchar_expr*이 NULL이면 NULL이 return됩니다.
- varchar 표현식이 하나의 매개변수로 주어지고 unichar 표현식이 다른 매개변수로 주어진 경우 varchar 표현식은 묵시적으로 unichar 표현식으로 변환되며 일부 truncation될 수 있습니다.

- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 patindex 함수를 실행할 수 있습니다.

참조 *함수* – charindex, substring

pagesize

설명 지정된 객체의 페이지 크기를 바이트로 return합니다.

구문

pagesize(*object_name* [, *index_name*]
or
pagesize(dbid, object_id [, index_id])

매개변수

object_name
찾을 객체의 이름입니다. 완전히 유효한 객체 이름을 지정하지 않으면 현재 데이터베이스가 검색됩니다.

index_name
검색에 사용하는 인덱스의 이름을 나타냅니다

dbid
*object_name*에 의해 지정된 데이터베이스의 ID입니다.

object_id
*object_name*에 의해 표시된 객체의 ID입니다.

index_id
*index_name*에 의해 표시된 인덱스의 ID입니다.

예제

예제 1

```
select pagesize(title, title_id)
from titles
```

titles 테이블의 title 열에 대한 pagesize를 선택합니다.

예제 2

```
select pagesize(4, )
```

titles 테이블(object_id 224000798)의 pagesize를 pubs2 데이터베이스(dbid 4)에서 선택합니다.

사용법

- *index_name*을 표시하지 않으면 테이블의 데이터 수준을 디폴트 값으로 사용합니다.

- 지정된 객체가 페이지 크기가 아닌 경우(예를 들어, 뷰의 이름이 제공된 경우) pagesize()는 0을 return합니다.
- 지정된 객체가 존재하지 않는 경우 pagesize()는 NULL을 return합니다.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자는 pagesize()를 실행할 수 있습니다.

pi

설명 상수 값 3.1415926535897936을 return합니다.

구문 pi()

매개변수 없음.

예제

```
select pi()

------------------
          3.141593
```

사용법

- pi는 수학 함수이며 상수 값 3.1415926535897931을 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 pi 함수를 실행할 수 있습니다.

참조 *함수* – degrees, radians

power

설명 지정된 수를 주어진 승수로 제곱한 값을 return합니다.

구문 power(*value*, *power*)

매개변수

value
숫자 값입니다.

power
정밀 숫자, 근사 숫자 또는 통화 값입니다.

예제

```
select power(2, 3)

-----------
          8
```

사용법

- power는 수학 함수이며 *value*를 *power* 승으로 제곱한 값을 return합니다. 결과 값은 *value*와 동일한 유형을 가집니다.

  numeric 또는 decimal 유형 표현식의 경우 결과 값은 77의 내부 정밀도(P) 및 표현식과 동일한 소수점 이하 자릿수를 가집니다.

- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 power 함수를 실행할 수 있습니다.

참조

함수 – exp, log

proc_role

설명

사용자가 지정된 롤(role)을 부여받았는지에 대한 정보를 return합니다.

구문

proc_role ("*role_name*")

매개변수

role_name
시스템 또는 사용자 정의 롤(role) 이름입니다.

예제

예제 1

```
create procedure sa_check as
if (proc_role("sa_role") > 0)
begin
     return(1)
end
print "You are a System Administrator."
```

사용자가 시스템 관리자인지 확인하는 프로시저를 작성합니다.

예제 2

```
select proc_role("sso_role")
```

사용자에게 시스템 보안 담당자 롤(role)이 부여되었는지 확인합니다.

예제 3

```
select proc_role("oper_role")
```

사용자에게 운영자 롤(role)이 부여되었는지 확인합니다.

사용법

- proc_role은 시스템 함수이며 지정된 롤(role)이 해당 사용자에게 부여되고 활성화되었는지 확인합니다.
- proc_role은 다음 중 하나가 참인 경우 0을 return합니다.
  - 지정된 롤(role)이 사용자에게 부여되지 않은 경우
  - 지정된 롤(role)을 포함하는 롤(role)이 사용자에게 부여되지 않은 경우
  - 지정된 롤(role)이 사용자에게 부여되었지만 활성화되지 않은 경우
- proc_role은 지정된 롤(role)이 해당 사용자에게 부여되고 활성된 경우 1을 return합니다.
- proc_role은 해당 사용자가 지정된 롤(role)을 포함하는 현재 활성화된 롤(role)을 가지고 있는 경우 2를 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 proc_role 함수를 실행할 수 있습니다.

참조

명령 – alter role, create role, drop role, grant, set, revoke

함수 – mut_excl_roles, role_contain, role_id, role_name, show_role

ptn_data_pgs

설명

partition 영역에서 사용한 데이터 페이지 수를 return합니다.

구문

ptn_data_pgs(*object_id*, *partition_id*)

매개변수

object_id

sysobjects, sysindexes 및 syspartitions의 id 열에 저장되는 테이블에 대한 객체 ID입니다.

partition_id
테이블의 partition 번호입니다.

예제

```
select ptn_data_pgs(object_id("salesdetail"), 1)

-----------
          5
```

사용법

- ptn_data_pgs는 시스템 함수이며 partition된 테이블의 데이터 페이지 수를 return합니다.
- 객체 ID를 얻으려면 object_id 함수를 사용하고 테이블의 partition 영역을 나열하려면 sp_helpartiton을 사용합니다.
- ptn_data_pgs가 return하는 데이터 페이지는 부정확할 수 있습니다. 가장 정확한 값을 얻으려면 ptn_data_pgs를 사용하기 전에 update partition statistics, dbcc checktable, dbcc checkdb 또는 dbcc checkalloc 명령을 사용하십시오.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

테이블 소유자만 ptn_data_pgs 함수를 실행할 수 있습니다.

참조

명령 – update partition statistics, dbcc

함수 – data_pgs, object_id

시스템 프로시저 – sp_helpartition

radians

설명

지정된 수에 대한 각의 크기를 라디안으로 return합니다.

구문

radians(*numeric*)

매개변수

numeric
정밀 숫자(numeric, dec, decimal, tinyint, smallint 또는 int), 근사 숫자(float, real 또는 double precision) 또는 money 유형의 열, 변수, 상수 표현식 또는 이들의 조합입니다.

예제

```
select radians(2578)

-----------
         44
```

사용법

- radians는 수학 함수이며 도(degree)를 라디안으로 변환합니다. 결과는 *numeric*과 동일한 유형을 가집니다.

  숫자 또는 Decimal 유형 표현식의 경우 결과는 77의 내부 정밀도(P) 및 숫자 표현식과 동일한 소수점 이하 자릿수를 가집니다.

  money 데이터 유형이 사용될 경우 float 유형으로 내부 변환을 수행하면 정밀도(P)를 잃을 수도 있습니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 radians 함수를 실행할 수 있습니다.

참조

함수 – degrees

rand

설명

지정된 시드 값을 사용하여 생성된 0과 1 사이의 임의 값을 return합니다.

구문

rand([*integer*])

매개변수

integer
모든 정수(tinyint, smallint 또는 int) 열 이름, 변수, 상수 표현식 또는 이들의 조합입니다.

예제

예제 1

```
select rand()

--------------------
            0.395740
```

예제 2

```
declare @seed int
select @seed=100
select rand(@seed)

--------------------
             0.000783
```

사용법

- rand는 수학 함수이며 선택적 정수를 시드 값으로 사용하여 0과 1 사이의 임의 부동 소수 값을 return합니다.

- rand 함수는 32비트 pseudo-random 정수 생성기의 출력을 사용합니다. 정수는 최대 32비트 정수로 나누어져 0.0과 1.0 사이의 double 값을 출력합니다. rand 함수는 서버 시작 시 시드 값이 임의로 주어지므로 처음에 상수 시드 값으로 이 함수를 초기화하지 않는 한 동일한 시퀀스의 임의 수를 얻기는 불가능합니다. rand 함수는 글로벌 자원입니다. rand 함수를 호출하는 여러 사용자는 단일 스트림의 pseudo-random 값에 따라 진행합니다. 반복 가능한 일련의 임의 수가 필요할 경우 사용자는 함수가 동일한 시드 값으로 시작되고 반복 시퀀스 동안 다른 사용자가 rand를 호출하지 않도록 해야 합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 rand 함수를 실행할 수 있습니다.

참조 *데이터 유형* – 근사 숫자 데이터 유형

replicate

설명 지정된 표현식을 주어진 횟수만큼 반복하여 이루어진 문자열을 return 합니다.

구문 replicate (*char_expr*|uchar_expr, *integer_expr*)

매개변수

char_expr
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 의 상수 표현식입니다.

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형의 상수 표현식입니다.

integer_expr
모든 정수(tinyint, smallint 또는 int) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

```
select replicate("abcd", 3)

------------
abcdabcdabcd
```

사용법

- replicate는 문자열 함수이며 지정된 횟수 또는 64K 공간 내에서 반복 가능한 최대 횟수만큼 반복된 동일한 표현식을 포함하는 *char_expr* 또는 *uchar_expr*과 동일한 데이터 유형의 문자열을 return합니다.
- *char_expr* 또는 *uchar_expr*이 NULL로서 단일 NULL을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 replicate 함수를 실행할 수 있습니다.

참조

함수 – stuff

reserved_pgs

설명

지정된 테이블이나 인덱스 및 내부 구조에 사용되는 보고서 페이지 등에 할당된 페이지의 수를 return합니다.

구문

reserved_pgs(*object_id*, {doampg|ioampg})

매개변수

object_id

테이블, 뷰 또는 기타 데이터베이스 객체에 대한 객체 ID인 숫자 표현식입니다. 객체 ID는 sysobjects의 id 열에 저장됩니다.

doampg | ioampg

테이블(doampg) 또는 인덱스(ioampg)를 지정합니다.

예제

```
select reserved_pgs(id, doampg)
from sysindexes where id =
    object_id("syslogs")

-------------
          534
```

syslogs 테이블에 대한 페이지 수를 return합니다.

사용법

- reserved_pgs는 시스템 함수입니다.
  - 테이블 또는 인덱스에 할당된 페이지 수를 return합니다.
  - 내부 구조에 사용된 페이지를 보고합니다.
  - 현재 데이터베이스의 객체에 대해서만 작동합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 reserved_pgs 함수를 실행할 수 있습니다.

참조 *명령* – update statistics

함수 – data_pgs

reverse

설명 지정된 문자열을 역순으로 나열하여 return합니다.

구문 reverse(*expression* | *uchar_expr*)

매개변수

expression

char, varchar, nchar, nvarchar, binary 또는 varbinary 유형의 문자, 이진 유형 열 이름, 변수 또는 상수 표현식입니다.

uchar_expr

unichar 또는 univarchar 유형의 문자, 이진 유형 열 이름, 변수 또는 상수 표현식입니다.

예제

예제1

```
select reverse("abcd")

----
dcba
```

예제 2

```
select reverse(0x12345000)

----------
0x00503412
```

사용법

- reverse는 문자열 함수이며 *expression*을 역순으로 return합니다.
- *expression*이 NULL이면 NULL을 return합니다.
- surrogate 쌍은 나눌 수 없는 것으로 처리되며 역순으로 될 수 없습니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 reverse 함수를 실행할 수 있습니다.

참조 *함수* – lower, upper

right

설명

표현식의 가장 오른쪽 부분에서 지정된 수만큼의 문자를 return합니다.

구문

right(*expression*, *integer_expr*)

매개변수

expression

char, varchar, nchar, unichar, nvarchar, univarchar, binary 또는 varbinary 유형의 문자, 이진 유형 열 이름, 변수 또는 상수 표현식입니다.

integer_expr

모든 정수(tinyint, smallint 또는 int) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

예제 1

```
select right("abcde", 3)

 ---
 cde
```

예제 2

```
select right("abcde", 2)

 --
 de
```

예제 3

```
select right("abcde", 6)

 -----
 abcde
```

예제 4

```
select right(0x12345000, 3)

 -------
 0x345000
```

예제 5

```
select right(0x12345000, 2)

 ------
 0x5000
```

예제 6

```
select right(0x12345000, 6)

 ---------
 0x12345000
```

사용법

- right는 문자열 함수이며 문자 또는 이진 표현식의 가장 오른쪽에서 지정된 수만큼의 문자를 return합니다.
- 지정된 가장 오른쪽의 부분이 두 번째 surrogate 쌍으로 시작하면 (낮은 surrogate 쌍) return 값은 다음 전체 문자로 시작됩니다. 따라서 하나 작은 문자가 return됩니다.
- return 값은 문자 또는 이진 표현식과 동일한 데이터 유형을 가집니다.
- *expression*이 NULL이면 NULL을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 right 함수를 실행할 수 있습니다.

참조

함수 – rtrim, substring

role_contain

설명

*role2*가 *role1*을 포함하는 경우 1을 return합니다.

구문

role_contain("role1", "role2")

매개변수

role1
시스템 또는 사용자 정의 롤(role) 이름입니다.

role2
다른 시스템 또는 사용자 정의 롤(role) 이름입니다.

예제

예제 1

```
select role_contain("intern_role", "doctor_role")

-----------
1
```

예제 2

```
select role_contain("specialist_role", "intern_role")
-----------
0
```

사용법

- role_contain은 시스템 함수이며 *role1*이 *role2*에 포함될 경우 1을 return합니다.
- 포함되는 롤(role) 및 롤(role) 계층에 대한 자세한 사항은 *시스템 관리 지침서*를 참조하십시오.
- 시스템 함수에 대한 자세한 사항은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 role_contain 함수를 실행할 수 있습니다.

참조

함수 – mut_excl_roles, proc_role, role_id, role_name

명령 – alter role

시스템 프로시저 – sp_activeroles, sp_displayroles, sp_role

role_id

설명

지정한 롤(role)의 시스템 롤(role) ID를 return합니다.

구문

role_id("*role_name*")

매개변수

role_name

시스템 또는 사용자 정의 롤(role) 이름입니다. 롤(role) 이름 및 롤(role) ID는 syssrvroles 시스템 테이블에 저장됩니다.

예제

예제 1

```
select role_id("sa_role")

------
0
```

sa_role의 시스템 롤(role) ID를 return합니다.

예제 2

```
select role_id("intern_role")

------
6
```

"intern_role"의 시스템 롤(role) ID를 return합니다.

사용법

- role_id는 시스템 함수이며 시스템 롤(role) ID(srid)를 return합니다. 시스템 롤(role) ID는 syssrvroles 시스템 테이블의 srid 열에 저장됩니다.
- *role_name*이 시스템의 유효한 롤(role)이 아닌 경우 Adaptive Server는 NULL을 return합니다.
- 롤(role)에 대한 자세한 내용은 *시스템 관리 지침서*를 참조하십시오.
- 시스템 함수에 대한 자세한 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 role_id 함수를 실행할 수 있습니다.

참조

함수 – mut_excl_roles, proc_role, role_contain, role_name

role_name

설명

지정한 시스템 롤(role) ID의 롤(role) 이름을 return합니다.

구문

role_name(*role_id*)

매개변수

role_id
롤(role)의 시스템 롤(role) ID(srid)입니다. 롤(role) 이름은 syssrvroles에 저장됩니다.

예제

```
select role_name(01)

------------------------------
sso_role
```

사용법

- role_name은 시스템 함수이며 롤(role) 이름을 return합니다.
- 시스템 함수에 대한 자세한 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 role_name 함수를 실행할 수 있습니다.

참조

함수 – mut_excl_roles, proc_role, role_contain, role_id

round

설명

지정된 수를 주어진 소수점 위치만큼 반올림한 값을 return합니다.

구문

round(*number*, *decimal_places*)

매개변수

number

모든 정밀 숫자(numeric, dec, decimal, tinyint, smallint 또는 int), 근사 숫자(float, real 또는 double precision) 또는 money 유형의 열, 변수, 상수 표현식 또는 이들의 조합입니다.

decimal_places

반올림할 소수점 위치입니다.

예제

예제 1

```
select round(123.4545, 2)

----------
  123.4500
```

예제 2

```
select round(123.45, -2)

----------
  100.00
```

예제 3

```
select round(1.2345E2, 2)

-----------------
       123.450000
```

예제 4

```
select round(1.2345E2, -2)

-----------------
       100.000000
```

사용법

- round는 수학 함수이며 *number*를 반올림하여 *decimal_places* 유효 자리를 갖도록 합니다.
- 양의 *decimal_places*는 유효 자리의 수가 소수점 오른쪽, 음의 *decimal_places*는 소수점 왼쪽임을 나타냅니다.
- 결과 값은 *number*와 동일한 유형을 가지고 숫자 및 Decimal 표현식의 경우 첫 번째 인자에 1을 더한 동일한 내부 정밀도(P) 및 *number*와 동일한 소수점 이하 자릿수를 가집니다.

- round는 항상 값을 return합니다. *decimal_places*가 음수이고 *number*의 유효 자릿수를 초과할 경우 Adaptive Server는 0을 return합니다. 0.00 형식으로 표현되며 소수점 오른쪽 0의 개수는 numeric의 소수점 이하 자릿수와 동일합니다. 예를 들면 다음과 같습니다.

  ```
  select round(55.55, -3)
  ```

  0.00 값을 return합니다.

- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 round 함수를 실행할 수 있습니다.

참조 *함수* – abs, ceiling, floor, sign, str

rowcnt

설명 지정된 테이블에서 행 수의 추정 값을 return합니다.

구문 rowcnt(sysindexes.doampg)

매개변수 *sysindexes.doampg*
– sysindexes에서 유지되는 행 수입니다.

예제

```
select name, rowcnt(sysindexes.doampg)
     from sysindexes
     where name in
           (select name from sysobjects
            where type = "U")

name
------------------------------ -----------
roysched                                87
salesdetail                            116
stores                                   7
discounts                                4
au_pix                                   0
blurbs                                   6
```

사용법

- rowcnt는 시스템 함수이며 테이블 행의 추정 수를 return합니다.
- Adaptive Server가 재부팅되고 트랜잭션이 복구될 경우 rowcnt에 의해 return되는 값이 달라질 수 있습니다. 값은 다음 명령 중 하나를 실행한 후에 가장 정확합니다.

- dbcc checkalloc
- dbcc checkdb
- dbcc checktable
- update all statistics
- update statistics

- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 rowcnt 함수를 실행할 수 있습니다.

참조

카탈로그 내장 프로시저 – sp_statistics

명령 – dbcc, update all statistics, update statistics

함수 – data_pgs

시스템 프로시저 – sp_helpartition, sp_spaceused

rtrim

설명 지정된 표현식의 후미 공백을 제거하여 return합니다.

구문 rtrim(*char_expr* | *uchar_expr*)

매개변수

char_expr
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형 등의 상수 표현식입니다.

예제

```
select rtrim("abcd    ")

--------
abcd
```

사용법

- rtrim은 문자열 함수이며 후미 공백을 제거합니다.
- Unicode의 경우 공백은 Unicode 값 U+0020으로 정의됩니다.
- *char_expr* 또는 *uchar_expr*이 NULL이면 NULL을 return합니다.

- 현재 문자 집합에서 공백 문자에 해당하는 값만 제거됩니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 − 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 rtrim 함수를 실행할 수 있습니다.

참조 *함수* − ltrim

6 장

함수: *show_role – valid_user*

show_role

설명

로그인의 현재 시스템 정의 롤(role)을 보여 줍니다.

구문

show_role()

매개변수

없음.

예제

예제 1

```
select show_role()

sa_role sso_role oper_role replication_role
```

예제 2

```
if charindex("sa_role", show_role()) >0
begin
     print "You have sa_role"
 end
```

사용법

- show_role은 시스템 함수이며 sa_role, sso_role, oper_role 또는 replication_role 등 로그인의 현재 시스템 정의 롤(role)을 return합니다. 로그인에 롤(role)이 없는 경우 show_role은 NULL을 return합니다.
- 데이터베이스 소유자가 setuser를 사용한 후 show_role을 실행하면 show_role은 setuser로 구현된 사용자가 아니라 해당 데이터베이스 소유자의 현재 롤(role)을 표시합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

show_role 함수는 모든 사용자가 실행할 수 있습니다.

참조

명령 – alter role, create role, drop role, grant, set, revoke

함수 – proc_role, role_contain

시스템 프로시저 – sp_activeroles, sp_displayroles, sp_role

show_sec_services

설명 해당 세션에서 현재 사용 중인 보안 서비스를 보여 줍니다.

구문 show_sec_services()

매개변수 없음.

예제

```
select show_sec_services()

encryption, replay_detection
```

사용자의 현재 세션에서 데이터 encryption과 재생 검색을 수행 중임을 보여 줍니다.

사용법

- 세션에서 사용 중인 보안 서비스를 표시하는 데 show_sec_services를 사용합니다.
- 보안 서비스가 활성화되지 않았으면 show_sec_services는 NULL을 return합니다.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 show_sec_services 함수를 실행할 수 있습니다.

참조 *함수* – is_sec_service_on

sign

설명 지정된 값의 기호(양수인 경우 +1, 0, 음수인 경우 -1)를 return합니다.

구문 sign(*numeric*)

매개변수

numeric
정밀 숫자(numeric, dec, decimal, tinyint, smallint 또는 int), 근사 숫자(float, real 또는 double precision) 또는 money 유형의 열, 변수, 상수 표현식 또는 이들의 조합입니다.

예제

예제1

```
select sign(-123)

-----------
         -1
```

예제 2

```
select sign(0)
```

```
-----------
          0
```

예제 3

```
select sign(123)

-----------
          1
```

사용법

- sign은 수학 함수이며 양의 값(+1), 영(0) 또는 음의 값(-1)을 return합니다.
- 결과 값은 숫자 표현식과 같은 유형으로서 같은 정밀도(P)와 소수점 이하 자릿수를 갖습니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 sign 함수를 실행할 수 있습니다.

참조

함수 – abs, ceiling, floor, round

sin

설명

라디안으로 지정된 각의 사인 값을 return합니다.

구문

sin(*approx_numeric*)

매개변수

approx_numeric
근사 숫자(float, real 또는 double precision) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

```
select sin(45)

--------------------
            0.850904
```

사용법

- sin은 수학 함수이며 라디안으로 지정된 각의 사인 값을 return합니다.
- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 sin 함수를 실행할 수 있습니다.

참조 *함수* – cos, degrees, radians

sortkey

설명 조합 동작에 따라 결과를 정렬하는 데 사용되는 값을 생성합니다. 이를 통해 라틴 문자 기반의 사전식 정렬 순서와 대/소문자 또는 액센트를 구분하는 디폴트 문자 집합과 관계없이 문자 조합 작업을 할 수 있습니다.

구문 sortkey (*char_expression*|uchar_expression) [, {*collation_name* | *collation_ID*}])

매개변수

char_expression
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

uchar_expression
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형의 상수 표현식입니다.

collation_name
사용할 조합을 지정하는 인용 문자열이나 문자 변수입니다. 표 6-1은 유효한 값을 보여 줍니다.

collation_ID
사용할 조합을 지정하는 정수 형태의 상수나 변수입니다. 표 6-1은 유효한 값을 보여 줍니다.

예제 **예제 1**

```
select * from cust_table where cust_name like "TI%" order by
(sortkey(cust_name, "dict")
```

유럽 언어 사전 순서에 의한 정렬을 보여 줍니다.

예제 2

```
select *from cust_table where cust name like "TI%" order by
(sortkey(cust-name, "gbpinyin")
```

간체 중국어 사전 순서에 의한 정렬을 보여 줍니다.

예제 3

인라인 옵션을 사용한 유럽 언어 사전 순서에 의한 정렬을 보여 줍니다.

```
select *from cust_table where cust_name like"TI%" order by cust_french_sort
```

예제 4

```
select * from cust_table where cust_name like "TI%" order by
cust_chinese_sort.
```

기존 키를 사용한 간체 중국어 사전 순서에 의한 정렬을 보여 줍니다.

사용법

- sortkey는 시스템 함수로서 조합 동작에 따라 결과를 정렬하는 데 사용되는 값을 생성합니다. 이를 통해 라틴 문자 기반의 사전식 정렬 순서와 대/소문자 또는 액센트를 구분하는 디폴트 문자 집합과 관계없이 문자 조합 작업을 할 수 있습니다. return 값은 sortkey 함수에서 얻어지는 입력 문자열의 코드화된 조합 정보가 들어 있는 varbinary 데이터 유형 값입니다.

  예를 들어, sortkey를 통해 return된 값은 원본 문자열과 함께 열에 저장할 수 있습니다. 문자 데이터를 원하는 순서로 검색하기 위해서는 sortkey의 실행 결과를 포함하는 열에 대한 order by 절을 select 문에 포함하기만 하면 됩니다.

  sortkey 함수를 사용하면 주어진 조합 기준 집합에 대한 return 값이 varbinary 데이터 유형에 대해 실행된 이진 비교에 적용됩니다.

  참고 sortkey는 각 입력 문자에 대해 최대 6바이트의 조합 정보를 생성할 수 있습니다. 그러므로 sortkey를 사용한 결과로 varbinary 데이터 유형의 255바이트 길이 제한을 초과할 수 있습니다. 이 경우 결과는 255바이트에 맞게 truncation되며 결과 문자열이 255바이트 미만이 될 때까지 각 입력 문자의 결과 바이트를 삭제합니다. 이 때 경고 메시지는 발생되지만 sortkey 함수를 포함한 쿼리나 트랜잭션 작업은 계속됩니다.

- *char_expression* 또는 *uchar_expression*은 서버의 디폴트 문자 집합에 인코딩된 문자로 이루어져야 합니다.
- *char_expression* 또는 *uchar_expression*은 빈 문자열일 수 있습니다. 빈 문자열인 경우에는 다음과 같이 수행됩니다.
  - sortkey는 길이가 0인 varbinary 값을 return합니다.
  - 빈 문자열에 공백을 저장합니다.

  빈 문자열은 데이터베이스 열의 NULL 문자열과는 다른 조합 값을 갖습니다.
- *char_expression* 또는 *uchar_expression*이 NULL이면 sortkey는 NULL 값을 return합니다.
- unicode 표현식에 지정된 정렬 순서가 없으면 Unicode 디폴트 정렬 순서가 사용됩니다.

- *collation_name* 또는 *collation_ID*의 값을 지정하지 않으면 sortkey는 이진 조합으로 가정합니다.

조합 테이블

다중 언어 정렬을 위해 사용할 수 있는 조합 테이블은 다음 두 가지 유형이 있습니다.

1 sortkey 함수에 의해 생성된 "내장된" 테이블. 이 함수는 ASE 11.5.1 이후의 모든 ASE 릴리스에 들어 있습니다. 내장 테이블을 지정하려면 조합 이름이나 조합 ID를 사용합니다.

2 Unilib 라이브러리 정렬 함수를 사용하는 외부 조합 테이블. 조합 이름을 사용하여 외부 테이블을 지정해야 합니다. 이들 *srt 파일은 $SYBASE/collate/unicode에 있습니다.

   이 두 가지 메소드는 모두 바르게 작동하나 "내장된" 테이블은 Sybase ASE 데이터베이스에 묶여 있으며 외부 테이블은 그렇지 않습니다. ASE 데이터베이스를 사용하는 경우 내장된 테이블이 최상의 성능을 제공합니다. 이 두 가지 메소드는 모두 영어, 유럽어 및 아시아어의 모든 혼합을 처리할 수 있습니다.

sortkey 사용 방법에는 다음 두 가지가 있습니다.

1 인라인(In-line) 방법: 인라인은 sortkey를 order by 절의 일부로 사용하며 기존 응용 프로그램을 개선하고 변경 사항을 최소화하는 데 유용합니다. 그러나 이 메소드는 정렬 키를 즉시 생성하며 따라서 1000개의 레코드가 넘는 대형 데이터 세트에 대해서는 최적의 성능을 제공하지 못합니다.

2 기존 키(Pre-existing key) 방법: 이 메소드는 새로운 고객 이름과 같이 다중 언어 정렬을 필요로 하는 새 레코드가 테이블에 추가될 때마다 sortkey를 호출합니다. 섀도우 열(binary 또는 varbinary 유형)은 데이터베이스 내에, 가능하면 동일한 테이블에 원하는 각 정렬 순서당 하나씩(예를 들어 불어, 중국어 등) 설정되어야 합니다. 쿼리가 출력 내용의 정렬을 요구하면 order by 절은 이들 섀도우 열 중 하나를 사용합니다. 이 메소드는 키가 이미 생성되고 저장되어 있으며 이진 값에만 기초하여 빠르게 비교되므로 최상의 성능을 보입니다.

사용 가능한 조합 규칙의 목록을 볼 수 있습니다. 내장 프로시저 sp_helpsort를 실행하거나 syscharsets에서 name, id 및 description을 쿼리하고 선택하여 목록을 인쇄합니다(type은 2003과 2999 사이임).

- 표 6-1에서는 *collation_name* 및 *collation_ID*에 대해 유효한 값을 나열합니다.

표 6-1: 조합 이름과 ID

설명	조합 이름	조합 ID
이진 정렬	binary	50
디폴트 Unicode 다국어	default	0
CP 850 대체, 액센트 비구분	altnoacc	39
CP 850 대체, 소문자 우선	altdict	45
CP 850 대체, 대/소문자 비구분	altnocsp	46
CP 850 스칸디나비아어 사전	scandict	47
CP 850 스칸디나비아어, 대/소문자 비구문	scannocp	48
GB 병음	gbpinyin	해당 사항 없음
Latin-1 영어, 불어, 독일어 사전	dict	51
Latin-1 영어, 불어, 독일어, 대/소문자 비구분	nocase	52
Latin-1 영어, 불어, 독일어, 대/소문자 비구분	nocasep	53
Latin-1 영어, 불어, 독일어, 액센트 비구분	noaccent	54
Latin-1 스페인어 사전	espdict	55
Latin-1 스페인어, 대/소문자 비구분	espnocs	56
Latin-1 스페인어, 액센트 비구분	espnoac	57
ISO 8859-5 키릴어 사전	cyrdict	63
ISO 8859-5 러시아어 사전	rusdict	58
ISO 8859-9 터키어 사전	turdict	72
Shift-JIS binary 순서	sjisbin	259
태국어 사전	thaidict	1

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 sortkey 함수를 실행할 수 있습니다.

참조 *함수* – compare

soundex

설명

표현식이 소리 나는 방식을 표시하는 4자의 코드를 return합니다.

구문

soundex(*char_expr* | *uchar_expr*)

매개변수

char_expr
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형의 상수 표현식입니다.

예제

```
select soundex ("smith"), soundex ("smythe")

----- -----
S530  S530
```

사용법

- soundex는 문자열 함수이며 연속적이고 유효한 단일 바이트 또는 더블 바이트의 로마 문자로 구성된 문자열에 대한 4자로 된 soundex 코드를 return합니다.
- soundex 함수는 알파 문자열을 4자리 코드로 변환하여 발음이 유사한 단어 또는 이름을 찾는 데 사용합니다. 모음은 문자열의 첫 번째 문자가 아닌 한 모두 무시됩니다.
- *char_expr* 또는 *uchar_expr*이 NULL이면 NULL을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 soundex 함수를 실행할 수 있습니다.

참조

함수 – difference

space

설명

지정된 숫자만큼의 1바이트 공간으로 이루어진 문자열을 return합니다.

구문

space(*integer_expr*)

매개변수

integer_expr
모든 정수(tinyint, smallint 또는 int) 유형의 열 이름, 변수 또는 상수 표현식입니다.

예제

```
select "aaa", space(4), "bbb"

--- ---- ---
aaa      bbb
```

사용법

- space는 문자열 함수이며 지정된 숫자만큼의 1바이트 공간을 가진 문자열을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 space 함수를 실행할 수 있습니다.

참조

함수 – isnull, rtrim

sqrt

설명

지정된 숫자의 제곱근을 return합니다.

구문

sqrt(*approx_numeric*)

매개변수

approx_numeric
근사 숫자(float, real 또는 double precision) 유형의 열 이름, 변수 또는 값이 양수인 상수 표현식입니다.

예제

```
select sqrt(4)

 2.000000
```

사용법

- sqrt는 수학 함수이며 지정된 값의 제곱근을 return합니다.
- 음수의 제곱근을 선택하면 Adaptive Server에서는 다음과 같은 에러 메시지가 나타납니다.

  ```
  Domain error occurred.
  ```

- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 sqrt 함수를 실행할 수 있습니다.

참조

함수 – power

str

설명

지정된 수의 해당 문자를 return합니다.

구문

str(*approx_numeric*[,*length* [,*decimal*]])

매개변수

approx_numeric
근사 숫자(float, real 또는 double precision) 유형의 열 이름, 변수 또는 상수 표현식입니다.

length
return될 문자 수를 설정합니다(소수점과 소수점 좌우의 모든 자릿수, 공백 등 포함). 디폴트는 10입니다.

decimal
return될 소수점 이하 자릿수를 설정합니다. 디폴트는 0입니다.

예제

예제 1

```
select str(1234.7, 4)

----
1235
```

예제 2

```
select str(-12345, 6)

------
-12345
```

예제 3

```
select str(123.45, 5, 2)

-----
123.5
```

사용법

- str은 문자열 함수이며 부동 소수점 수의 문자 표현 값을 return합니다. 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.
- *length*와 *decimal*은 선택 사항입니다. 사용된 경우에는 음수가 아니어야 합니다. str에서는 지정된 길이에 맞도록 소수점 부분을 반올림합니다. 길이는 소수점까지 포함해야 하고 음수인 경우에는 기호까지 포함해야 합니다. 결과 값의 소수 부분은 지정된 길이에 맞도록 반올림합니다. 그러나 정수 부분이 지정된 길이와 맞지 않을 경우 str은 지정된 길이만큼 별표를 표시해서 return합니다. 예를 들면 다음과 같습니다.

```
select str(123.456, 2, 4)
--
**
```

*approx_numeric*이 짧은 경우에는 지정된 길이에서 오른쪽에 자리 맞춤하고 *approx_numeric*이 긴 경우에는 지정된 소수 자리 수에 맞게 truncation됩니다.

- *approx_numeric*이 NULL이면 NULL을 return합니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 str 함수를 실행할 수 있습니다.

참조

함수 – abs, ceiling, floor, round, sign

stuff

설명

한 문자열에서 지정된 수만큼의 문자를 삭제하고 이를 다른 문자열로 바꾼 문자열을 return합니다.

구문

stuff(*char_expr1*|*uchar_expr1*, *start*, *length*, *char_expr2*|*uchar_expr2*)

매개변수

char_expr1
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다.

uchar_expr1
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형의 상수 표현식입니다.

start
삭제를 시작할 문자 위치를 지정합니다.

length
삭제할 문자 수를 지정합니다.

char_expr2
다른 문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형의 상수 표현식입니다.

uchar_expr2
다른 문자 유형 열 이름, 변수 또는 unichar 또는 univarchar 유형의 상수 표현식입니다.

예제

예제 1

```
select stuff("abc", 2, 3, "xyz")

----
axyz
```

예제 2

```
select stuff("abcdef", 2, 3, null)

go
---
aef
```

예제 3

```
select stuff("abcdef", 2, 3, "")

----
a ef
```

사용법

- stuff는 문자열 함수로서 *start*에서 *char_expr1* 또는 *uchar_expr1*로부터 *length* 문자를 삭제한 후 *start*에서 *char_expr2* 또는 *uchar_expr2*를 *char_expr1* 또는 *uchar_expr2*에 삽입합니다. 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.
- 시작 위치 또는 길이가 음수일 경우 NULL 문자열이 return됩니다. 시작 위치가 *expr1*보다 길면 NULL 문자열이 return됩니다. 삭제할 길이가 *expr1*보다 길면 *expr1*의 마지막 문자까지 삭제됩니다(예제 1 참조).
- 시작 위치가 surrogate 쌍의 중간에 오면 시작은 하나 작게 조정되며 시작 길이 위치가 surrogate 쌍의 중간에 오면 길이는 하나 작게 조정됩니다.
- stuff를 사용하여 문자를 삭제하려면 *expr2*를 빈 따옴표보다는 "NULL"로 대체합니다. null 문자를 지정하기 위해 " "를 사용하면 공백으로 바뀝니다(예제 2와 예제 3 참조).
- *char_expr1* 또는 *uchar_expr1*이 NULL이면 NULL을 return합니다. *char_expr1* 또는 *uchar_expr1*이 문자열 값이고 *char_expr2* 또는 *uchar_expr2*가 NULL이면 삭제된 문자는 다른 문자로 바뀌지 않습니다.
- varchar 표현식이 하나의 매개변수로 주어지고 unichar 표현식이 다른 매개변수로 주어지면 varchar 표현식은 묵시적으로 unichar(truncation될 수 있음)로 변환됩니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 stuff 함수를 실행할 수 있습니다.

참조 *함수* – replicate, substring

substring

설명 다른 문자열로부터 지정된 수만큼의 문자열을 추출하여 만든 문자열을 return합니다.

구문 substring(*expression*, *start*, *length*)

매개변수

expression
이진 또는 문자 유형의 열 이름 변수 또는 상수 표현식입니다. char, nchar, unichar, varchar, univarchar 또는 nvarchar 데이터, binary 또는 varbinary 등이 올 수 있습니다.

start
substring이 시작되는 문자의 위치를 지정합니다.

length
substring의 문자 수를 지정합니다.

예제

예제 1

```
select au_lname, substring(au_fname, 1, 1)
from authors
```

예를 들어 "Bennet A"처럼 만든 사람의 성과 첫 번째 이니셜을 표시합니다.

예제 2

```
select substring(upper(au_lname), 1, 3)
from authors
```

만든 사람의 성을 대문자로 변환하고 처음 세 글자를 표시합니다.

예제 3

```
select substring((pub_id + title_id), 1, 6)
from titles
```

pub_id와 title_id를 연결하고 그 결과 문자열의 처음 여섯 글자를 표시합니다.

예제 4

```
select substring(xactid,5,2)
from syslogs
```

각 위치가 두 개의 이진수로 표현되는 이진 필드로부터 하위 네 개 숫자를 추출합니다.

사용법

- substring은 문자열 함수이며 문자 또는 이진 문자열의 일부를 return합니다. 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.
- substring의 인자 중 일부가 NULL이면 substring은 NULL을 return합니다.
- uchar_expr1의 시작에서 시작 위치가 surrogate 쌍에 중간에 오면 시작은 하나가 작게 조정됩니다. uchar_expr1의 시작에서 시작 길이 위치가 surrogate 쌍에 중간에 오면 길이는 하나가 작게 조정됩니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 substring 함수를 실행할 수 있습니다.

참조

함수 – charindex, patindex, stuff

sum

설명

값의 합계를 return합니다.

구문

sum([all | distinct] *expression*)

매개변수

all
sum을 모든 값에 적용합니다. all이 디폴트입니다.

distinct
sum을 적용하기 전에 중복되는 값을 제거합니다. distinct는 선택 사항입니다.

expression
열 이름, 상수, 함수 그리고 산술 또는 비트 단위 연산자나 서브쿼리로 연결된 열 이름, 상수, 함수의 모든 조합입니다. 총계적으로 표현식은 열 이름인 경우가 많습니다. 자세한 내용은 179 페이지의 "표현식"을 참조하십시오.

예제

예제 1

```
select avg(advance), sum(total_sales)
from titles
where type = "business"
```

모든 비즈니스 관련 도서의 총 매출 합계와 평균 선금을 계산합니다. 이들 집계 함수(aggregate function) 각각은 검색한 모든 행을 대상으로 계산한 하나의 합계 값을 산출합니다.

예제 2

```
select type, avg(advance), sum(total_sales)
from titles
group by type
```

group by 절과 함께 사용하면 집계 함수(aggregate function)는 전체 테이블이 아니라 각 그룹에 대해서 단일 값을 산출합니다. 이 구문은 각 유형의 도서에 대한 합계 값을 산출합니다.

예제 3

```
select pub_id, sum(advance), avg(price)
from titles
group by pub_id
having sum(advance) > $25000 and avg(price) > $15
```

titles 테이블을 출판사별로 그룹화한 후 총 선금 지급이 $25,000을 넘고 도서의 평균 가격이 $15를 넘는 출판사 그룹만 포함시킵니다.

사용법

- sum은 집계 함수(aggregate function)이며 한 열에 있는 모든 값의 합계를 구합니다. sum은 숫자(integer, floating point 또는 money) 데이터 유형에만 사용할 수 있습니다. 합계 계산에서 NULL 값은 무시됩니다.
- 집계 함수(aggregate function)에 대한 일반적인 사항은 45 페이지의 "집계 함수(aggregate function)"를 참조하십시오.
- 정수 데이터를 합산할 때, 해당 열의 데이터 유형이 smallint 또는 tinyint인 경우에도 Adaptive Server는 이를 int 값으로 처리합니다. DB-Library 프로그램에서 오버플로 에러를 피하려면 평균 또는 합계의 결과 값에 대한 모든 변수를 int 유형으로 선언하십시오.
- sum은 이진 데이터 유형에는 사용할 수 없습니다.
- 이 함수는 숫자 유형만 정의하므로 Unicode 표현식은 에러를 발생합니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 sum 함수를 실행할 수 있습니다.

참조

명령 – compute **절**, group by **및** having **절**, select, where **절**

함수 – count, max, min

suser_id

설명

syslogins 테이블로부터 서버 사용자의 ID 번호를 return합니다.

구문

suser_id([*server_user_name*])

매개변수

server_user_name
Adaptive Server의 로그인 이름입니다.

예제

예제 1

```
select suser_id()

------
     1
```

예제 2

```
select suser_id("margaret")

------
     5
```

사용법

- suser_id는 시스템 함수이며 syslogins에서 서버 사용자의 ID 번호를 return합니다. 시스템 함수에 대한 일반적인 사항은 64 페이지의 "시스템 함수"를 참조하십시오.
- sysusers 테이블로부터 특정 데이터베이스에 있는 사용자의 ID 번호를 찾으려면 user_id 시스템 함수를 사용합니다.
- 입력된 *server_user_name*이 없으면 suser_id는 현재 사용자의 서버 ID를 return합니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 suser_id 함수를 실행할 수 있습니다.

참조

함수 – suser_name, user_id

suser_name

설명

현재 서버 사용자 또는 서버 ID가 지정된 사용자의 이름을 return합니다.

구문

suser_name([*server_user_id*])

매개변수

server_user_name
Adaptive Server 사용자 ID입니다.

예제

예제 1

```
select suser_name()

------------------------------
sa
```

예제 2

```
select suser_name(4)

------------------------------
margaret
```

사용법

- suser_name은 시스템 함수이며 해당 서버 사용자의 이름을 return합니다. 서버 사용자 ID는 syslogins에 저장됩니다. 입력된 *server_user_id*가 없으면 suser_name은 현재 사용자의 이름을 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 suser_name 함수를 실행할 수 있습니다.

참조

함수 – suser_id, user_name

syb_sendmsg

설명

메시지를 UDP(사용자 데이터그램 프로토콜) 포트로 보냅니다.

구문

syb_sendmsg *ip_address*, *port_number*, *message*

매개변수

ip_address
UDP 응용 프로그램이 실행되는 컴퓨터의 IP 주소입니다.

port_number
UDP 포트의 포트 번호입니다.

message
전송할 메시지입니다. 최고 길이는 255자입니다.

예제

예제 1

```
select syb_sendmsg("120.10.20.5", 3456, "Hello")
```

IP 주소 120.10.20.5의 3456 포트로 "Hello" 메시지를 보냅니다.

예제 2

```
declare @msg varchar(255)
select @msg = "Message to send"
select syb_sendmsg (ip_address, portnum, @msg)
from sendports
where username = user_name()
```

사용자 테이블에서 IP 주소와 포트 번호를 읽고 전송할 메시지에 대해 변수를 사용합니다.

사용법

- syb_sendmsg는 Windows NT에서 지원되지 않습니다.
- UDP 메시징을 활성화하려면 시스템 보안 담당자가 구성 매개변수 allow sendmsg를 1로 설정해야 합니다.
- syb_sendmsg에는 보안 검사가 수행되지 않습니다. syb_sendmsg를 사용하여 네트워크를 통해 중요한 정보를 보낼 때는 반드시 주의를 기울여야 합니다. 이 기능을 사용함으로써 발생하는 모든 보안 문제는 사용자의 책임입니다.
- UDP 포트를 생성하는 C 프로그램 예제에 대한 사항은 sp_sendmsg를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 syb_sendmsg 함수를 실행할 수 있습니다.

참조

시스템 프로시저 – sp_sendmsg

tan

설명

라디안으로 지정된 각의 탄젠트 값을 return합니다.

구문

tan(*angle*)

매개변수

angle

라디안 각의 크기로서 열 이름, 변수 또는 float, real, double precision 등의 유형 또는 이 유형들 중 하나로 변환할 수 있는 모든 데이터 유형으로 표현됩니다.

예제

```
select tan(60)

--------------------
          0.320040
```

사용법

- tan은 수학 함수이며 라디안으로 지정된 각의 탄젠트 값을 return합니다.

- 수학 함수에 대한 일반적인 사항은 60 페이지의 "수학 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 tan 함수를 실행할 수 있습니다.

참조 *함수* – atan, atn2, degrees, radians

textptr

설명 포인터를 text 또는 image 열의 첫 페이지로 return합니다.

구문 textptr(*column_name*)

매개변수

column_name
text 열의 이름입니다.

예제

예제 1

```
declare @val binary(16)
select @val = textptr(copy) from blurbs
where au_id = "486-29-1786"
readtext blurbs.copy @val 1 5
```

이 예제에서는 textptr 함수를 사용하여 저자의 *blurbs* 테이블에 있는 *au_id* 486-29-1786과 연결된 *copy*라는 text 열의 위치를 찾습니다. 이 텍스트 포인터는 로컬 변수 *@val*에 설정되어 있고 readtext 명령의 매개변수로 제공되어 두 번째 바이트(오프셋 1)에서 시작하여 5바이트를 return합니다.

예제 2

```
select au_id, textptr(copy) from blurbs
```

blurbs 테이블에서 *title_id* 열과 *copy* 열의 16바이트 텍스트 포인터를 표시합니다.

사용법

- textptr은 텍스트 및 이미지 함수이며 텍스트 포인터 값인 16바이트 varbinary 값을 return합니다.
- text 또는 image 열이 null이 아닌 insert 또는 update 구문에 의해 초기화되지 않은 경우 textptr은 NULL 포인터를 return합니다. 텍스트 포인터가 존재하는지 검사하려면 textvalid를 사용합니다. 유효한 텍스트 포인터 없이 writetext 또는 readtext를 사용할 수는 없습니다.

- 텍스트 및 이미지 함수에 대한 일반적인 사항은 65 페이지의 "텍스트 및 이미지 함수"를 참조하십시오.

참고 varbinary 값의 후미 f는 값이 테이블에 저장될 때 truncation 됩니다. text 포인터 값을 테이블에 저장하는 경우 binary를 해당 열의 데이터 유형으로 사용합니다.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 textptr 함수를 실행할 수 있습니다.

참조 *데이터 유형* – text 및 image 데이터 유형

함수 – textvalid

textvalid

설명 지정된 text 열에 대한 포인터가 유효하면 1을 return하고 유효하지 않으면 0을 return합니다.

구문 textvalid("*table_name.column_name*", *textpointer*)

매개변수 "*table_name.column_name*"
테이블과 text 열의 이름입니다.

textpointer
텍스트 포인터 값입니다.

예제

```
select textvalid ("texttest.blurb", textptr(blurb))
from texttest
```

texttest 테이블의 blurb 열의 각 값에 유효한 텍스트 포인터가 존재하는지 보고합니다.

사용법

- textvalid는 텍스트 및 이미지 함수이며 주어진 텍스트 포인터가 유효한지 확인합니다. 포인터가 유효하면 1을 return하고 유효하지 않으면 0을 return합니다.
- text 또는 image 열에 대한 식별자는 테이블 이름을 포함해야 합니다.
- 텍스트 및 이미지 함수에 대한 일반적인 사항은 65 페이지의 "텍스트 및 이미지 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 textvalid 함수를 실행할 수 있습니다.

참조 *데이터 유형* – text 및 image 데이터 유형

함수 – textptr

to_unichar

설명 정수 표현식의 값을 취하면서 unichar 표현식을 return합니다.

구문 to_unichar (integer_expr)

매개변수 *integer_expr*
모든 정수(tinyint, smallint 또는 int) 유형의 열 이름, 변수 또는 상수 표현식입니다.

사용법

- to_unichar는 문자열 함수로서 Unicode 정수 값을 Unicode 문자 값으로 변환합니다.
- unichar 표현식이 surrogate 쌍의 반만 참조하는 경우 에러 메시지가 나타나고 연산이 중지됩니다.
- *integer_expr*이 NULL이면 NULL을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 to_unichar를 실행할 수 있습니다.

참조 *데이터 유형* – text 및 image 데이터 유형

함수 – char

tsequal

설명 찾아보기를 위해 선택된 이후 변경된 행에서 업데이트가 이루어지는 것을 방지하기 위해 timestamp 값을 비교합니다.

구문 tsequal(*browsed_row_timestamp*, *stored_row_timestamp*)

매개변수 *browsed_row_timestamp*
찾아본 행의 timestamp 열입니다.

stored_row_timestamp
저장된 행의 timestamp 열입니다.

예제

```
update publishers
set city = "Springfield"
where pub_id = "0736"
and tsequal(timestamp, 0x0001000000002ea8)
```

현재 버전의 publishers 테이블에서 timestamp 열을 검색하여 timestamp 열에 있는 저장된 값과 비교합니다. 두 timestamp 열에 있는 값이 동일하면 해당 행을 업데이트합니다. 이 값이 일치하지 않으면 에러 메시지를 return합니다.

사용법

- tsequal은 시스템 함수이며 찾아보기를 위해 선택된 이후 변경된 행에서 업데이트가 이루어지는 것을 방지하기 위해 timestamp 열의 값들을 비교합니다. 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.
- tsequal 함수를 사용하면 DB-Library의 dbqual 함수를 호출하지 않고 찾아보기 모드를 이용할 수 있습니다. 찾아보기 모드에서는 데이터를 보면서 업데이트를 수행할 수 있는 기능을 지원하며 호스트 프로그래밍 언어와 Open Client를 사용하는 프론트엔드 응용 프로그램에서 사용됩니다. 행에 타임스탬프 표시가 되어 있는 테이블은 찾아보기를 수행할 수 있습니다.
- 프론트엔드 응용 프로그램에서 테이블을 찾아보려면 Adaptive Server로 전송하는 select 구문의 끝에 for browse 키워드를 추가합니다. 예를 들면 다음과 같습니다.

```
Start of select statement in an Open Client application
...
   for browse

Completion of the Open Client application routine
```

- tsequal 함수는 select 구문의 where 절에서는 사용하지 않아야 하며 where 절의 나머지 부분이 단일 고유 행에 대응하는 insert 및 update 구문의 where 절에서만 사용해야 합니다.

  timestamp 열이 검색 절로 사용되는 경우 timestamp1 = timestamp2와 같이 일반적인 varbinary 열처럼 비교되어야 합니다.

찾아보기를 위한 새로운 테이블의 타임스탬프 표시

- 사용할 새로운 테이블을 생성해서 찾아보기에 사용하려면 이름이 timestamp인 열을 테이블 정의에 포함합니다. 이 열에는 timestamp 데이터 유형이 자동으로 할당되므로 데이터 유형을 따로 지정할 필요가 없습니다. 예를 들면 다음과 같습니다.

```
create table newtable(col1 int, timestamp,
    col3 char(7))
```

행을 삽입 또는 업데이트할 때마다 Adaptive Server는 고유한 varbinary 값을 timestamp 열에 자동으로 할당하여 시간 표시를 합니다.

기존 테이블의 타임스탬프 표시

- 기존 테이블을 찾아보기에 사용하려면 alter table을 사용하여 이름이 timestamp인 열을 추가합니다. 예를 들면 다음과 같습니다.

  ```
  alter table oldtable add timestamp
  ```

  기존의 각 행에 NULL 값을 가진 timestamp 열을 추가합니다. 타임스탬프를 생성하려면 새로운 열 값을 지정하지 않고 기존의 각 행을 업데이트합니다. 예를 들면 다음과 같습니다.

  ```
  update oldtable
  set col1 = col1
  ```

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 tsequal 함수를 실행할 수 있습니다.

참조 *데이터 유형* – 타임스탬프 데이터 유형

uhighsurr

설명 *start* 위치의 Unicode 값이 surrogate 쌍의 상반부인 경우(쌍의 처음에 나타남) 1을 return하고 그렇지 않은 경우 0을 return합니다.

구문 uhighsurr(*uchar_expr*,start)

매개변수

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형 등의 상수 표현식입니다.

start
조사하려는 문자 위치를 지정합니다.

사용법

- uhighsurr은 문자열 함수로서 대신 처리를 위한 명시적 코드를 작성할 수 있게 합니다. 특히 uhighsurr()이 true인 Unicode 문자에서 substring이 시작할 경우 적어도 2개의 Unicode 값의 substring을 추출해야 합니다(*substr*은 surrogate 쌍의 반을 추출하지 않음).
- *uchar_expr*이 NULL이면 NULL을 return합니다.

- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 uhighsurr 함수를 실행할 수 있습니다.

참조 *함수* – ulowsurr

ulowsurr

설명 *start* 위치에서 Unicode 값이 surrogate 쌍의 하반부인 경우(쌍의 두 번째에 나타남) 1을 return하고 그렇지 않은 경우 0을 return합니다.

구문 ulowsurr(*uchar_expr*,start)

매개변수

uchar_expr
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형 등의 상수 표현식입니다.

start
조사하려는 문자 위치를 지정합니다.

사용법

- ulowsurr은 문자열 함수로서 substr(), stuff() 및 right()가 수행하는 조정 부분의 명시적 코드를 작성할 수 있게 합니다. 특히 substring이 ulowsurr()이 true인 Unicode 값에서 끝날 경우 사용자는 1이 적은 문자(또는 1이 많은)의 substring을 추출해야 하는 것을 알게 됩니다. substr()은 일치하지 않는 surrogate 쌍을 포함하는 문자열을 추출하지 않습니다.
- *uchar_expr*이 NULL이면 NULL을 return합니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 ulowsurr 함수를 실행할 수 있습니다.

참조 *함수* – uhighsurr

upper

설명 지정된 문자열의 대문자 형태를 return합니다.

구문 upper(*char_expr*)

매개변수 *char_expr*
문자 유형 열 이름, 변수 또는 char, unichar, varchar, nchar, nvarchar 또는 univarchar 유형 등의 상수 표현식입니다.

예제

```
select upper("abcd")

----
ABCD
```

사용법

- upper는 문자열 함수이며 소문자를 대문자로 변환하여 문자 값을 return합니다.
- *char_expr* 또는 *uchar_expr*이 NULL이면 NULL을 return합니다.
- 대문자가 없는 문자는 수정되지 않습니다.
- unichar 표현식이 surrogate 쌍의 반만 포함하여 생성되면 에러 메시지가 표시되고 연산이 중지됩니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 upper 함수를 실행할 수 있습니다.

참조 *함수* – lower

uscalar

설명 표현식 내의 첫 Unicode 문자에 대한 Unicode 스칼라 값을 return합니다.

구문 uscalar(u*char_expr*)

매개변수 *uchar_expr*
문자 유형 열 이름, 변수 또는 unichar, univarchar 유형 등의 상수 표현식입니다.

예제

사용법

- uscalar는 문자열 함수로서 표현식 내의 첫 Unicode 문자에 대한 Unicode 값을 return합니다.
- *uchar_expr*이 NULL이면 NULL을 return합니다.
- 일치하지 않는 surrogate 쌍의 반을 포함한 *uchar_expr*에 uscalar가 호출되면 에러가 일어나고 연산이 중지됩니다.
- 문자열 함수에 대한 일반적인 내용은 62 페이지의 "문자열 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 uscalar 함수를 실행할 수 있습니다.

참조

함수 – ascii

used_pgs

설명

테이블이나 인덱스가 사용하는 페이지의 수를 return합니다. clustered 인덱스가 있는 all-pages-locked 테이블의 경우 테이블과 인덱스 페이지의 합을 return합니다.

구문

used_pgs(*object_id, doampg, ioampg*)

매개변수

object_id
사용된 페이지를 보는 테이블의 객체 ID입니다. 인덱스가 사용한 페이지를 보려면 인덱스가 속한 테이블의 객체 ID를 지정합니다.

doampg
sysindexes의 doampg 열에 저장된 clustered 인덱스 또는 테이블의 객체 할당 맵(OAM)에 대한 페이지 수입니다.

ioampg
sysindexes의 ioampg 열에 저장된 nonclustered 인덱스의 할당 맵의 페이지 수입니다.

예제

예제 1

```
select name, id, indid, doampg, ioampg
from sysindexes where id = object_id("titles")

name          id          indid  doampg   ioampg
------------ ----------- ------ -------- -------
titleidind      208003772      1      560     552
titleind        208003772      2        0     456

select used_pgs(208003772, 560, 552)
```

```
-----------
          6
```

titles 테이블의 clustered 인덱스와 데이터가 사용한 페이지 수를 return합니다.

예제 2

```
select name, id, indid, doampg, ioampg
from sysindexes where id = object_id("stores")

name          id          indid  doampg   ioampg
------------- ----------- ------ -------- -------
stores          240003886      0       464       0

select used_pgs(240003886, 464, 0)

-----------
          2
```

인덱스가 없는 stores 테이블이 사용한 페이지 수를 return합니다.

사용법

- used_pgs는 시스템 함수로서 다음을 return합니다.
  - clustered 인덱스가 있는 all-pages-locked 테이블의 경우 테이블과 인덱스 페이지의 합
  - clustered 인덱스가 없는 테이블과 data-only-locked 테이블의 경우 테이블에서 사용된 페이지의 수
  - data-only-locked 테이블 상의 clustered 인덱스와 nonclustered 인덱스의 경우 인덱스 내의 페이지의 수
- 위 예제에서 indid 0은 테이블을 의미하며 indid 1은 clustered 인덱스를, 2-250의 indid는 nonclustered 인덱스를, indid 255는 text 또는 image 데이터를 의미합니다.
- used_pgs 함수는 현재 데이터베이스의 객체에 대해서만 작용합니다.
- 각 테이블과 테이블의 각 인덱스에는 OAM(객체 할당 맵)이 있으며 여기에는 객체에 할당되고 사용된 페이지 수에 대한 정보가 들어 있습니다. 이 정보는 페이지가 할당 또는 할당 해제될 때 대부분의 Adaptive Server 프로세스에서 업데이트됩니다. sp_spaceused 시스템 프로시저는 이 값을 읽어서 간단한 공간 추정값을 제공합니다. 일부 dbcc 명령은 일관성 검사를 수행하는 동안 이 값들을 업데이트합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 used_pgs 함수를 실행할 수 있습니다.

참조 *함수* – data_pgs, object_id

user

설명 현재 사용자의 이름을 return합니다.

구문 user

매개변수 없음.

예제

```
select user

------
dbo
```

사용법

- upper는 시스템 함수이며 사용자의 이름을 return합니다.
- sa_role이 현재 사용 중이면 사용자는 사용 중인 모든 데이터베이스에서 자동으로 데이터베이스 소유자가 됩니다. 데이터베이스 내부에서 데이터베이스 소유자의 사용자 이름은 항상 "dbo"입니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준 SQL92 – 호환성 수준: Transact-SQL 확장

권한 모든 사용자가 user 함수를 실행할 수 있습니다.

참조 *함수* – user_name

user_id

설명 지정된 사용자 또는 데이터베이스의 현재 사용자의 ID 번호를 return합니다.

구문 user_id([*user_name*])

매개변수

user_name
사용자의 이름입니다.

예제

예제 1

```
select user_id()

------
     1
```

예제 2

```
select user_id("margaret")

------
     4
```

사용법

- user_id는 시스템 함수이며 사용자의 ID 번호를 return합니다. 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.
- user_id는 현재 데이터베이스의 sysusers로부터 이 번호를 보고합니다. 입력된 *user_name*이 없으면 user_id는 현재 사용자의 ID를 return합니다. Adaptive Server의 모든 데이터베이스에서 번호가 동일한 서버 사용자 ID를 찾으려면 suser_id 함수를 사용합니다.
- 데이터베이스 내부에서 "guest" 사용자의 ID는 항상 2입니다.
- 데이터베이스 내부에서 데이터베이스 소유자의 user_id는 항상 1입니다. 현재 sa_role을 가지고 있는 사람은 사용 중인 모든 데이터베이스에서 자동으로 데이터베이스 소유자가 됩니다. 실제 사용자 ID로 돌아가려면 user_id를 실행하기 전에 set sa_role off를 사용하십시오. 데이터베이스에서 유효한 사용자가 아닌 경우에 set sa_role off를 사용하면 Adaptive Server에서 에러 메시지를 return합니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

자신의 것이 아닌 user_name에서 이 함수를 사용하려면 시스템 관리자 또는 시스템 보안 담당자여야 합니다

참조

명령 – setuser

함수 – suser_id, user_name

user_name

설명

해당 데이터베이스 내부에서 지정된 사용자 또는 현재 사용자의 이름을 return합니다.

구문

user_name([*user_id*])

매개변수

user_id
사용자의 ID입니다.

예제

예제 1

```
select user_name()

------------------------------
 dbo
```

예제 2

```
select user_name(4)

------------------------------
margaret
```

사용법

- user_name은 시스템 함수이며 현재 데이터베이스의 사용자 ID를 기준으로 사용자의 이름을 return합니다. 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.
- 입력된 *user_id*가 없으면 user_name은 현재 사용자의 이름을 return합니다.
- sa_role이 현재 사용 중이면 사용자는 사용 중인 모든 데이터베이스에서 자동으로 데이터베이스 소유자가 됩니다. 데이터베이스 내부에서 데이터베이스 소유자의 user_name은 항상 "dbo"입니다.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

자신의 것이 아닌 user_id에서 이 함수를 사용하려면 시스템 관리자 또는 시스템 보안 담당자여야 합니다.

참조

함수 – suser_name, user_id

valid_name

설명

지정된 문자열이 유효한 식별자가 아니면 0을 return하고 유효한 식별자인 경우에는 0 이외의 숫자를 return합니다.

구문

valid_name(*character_expression*)

매개변수

character_expression
문자 유형 열 이름, 변수 또는 char, varchar, nchar, nvarchar 유형 등의 상수 표현식입니다. 상수 표현식은 반드시 따옴표를 사용해야 합니다.

예제

```
create procedure chkname
@name varchar(30)
as
     if valid_name(@name) = 0
     print "name not valid"
```

식별자가 유효한지 확인할 수 있는 프로시저를 작성합니다.

사용법

- valid_name은 시스템 함수이며 *character_ expression*이 유효한 식별자가 아닌 경우(즉 에러 문자, 30바이트가 넘는 경우 또는 예약어인 경우)에는 0을 return하고 유효한 경우에는 0 이외의 숫자를 return합니다.
- Adaptive Server 식별자는 단일 바이트 문자 또는 다중 바이트 문자 사용 여부와 상관없이 최고 30바이트까지 길이가 허용됩니다. 식별자의 첫 번째 문자는 현재 문자 집합의 정의에 따라 반드시 알파벳이거나 밑줄 친(_) 문자여야 합니다. 파운드 기호(#)로 시작하는 임시 테이블 이름, @ 기호로 시작하는 로컬 변수 이름 등은 이 규칙에서 예외입니다. valid_name 함수는 파운드 기호(#)와 @ 기호로 시작하는 식별자에 대해서는 0을 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

모든 사용자가 valid_name 함수를 실행할 수 있습니다.

참조

시스템 프로시저 – sp_checkreswords

valid_user

설명

지정된 ID가 해당 Adaptive Server의 데이터베이스 중 최소한 하나에서 유효한 사용자 또는 가명(alias)일 경우 1을 return합니다.

구문

valid_user(*server_user_id*)

매개변수

server_user_id

서버 사용자 ID입니다. 서버 사용자 ID는 syslogins의 suid 열에 저장됩니다.

예제

```
select valid_user(4)

---------------
              1
```

사용법

- valid_user는 시스템 함수이며 지정된 ID가 해당 Adaptive Server의 데이터베이스 중 최소한 하나 이상에서 유효한 사용자 또는 가명(alias)일 경우 1을 return합니다.
- 시스템 함수에 대한 일반적인 내용은 64 페이지의 "시스템 함수"를 참조하십시오.

표준

SQL92 – 호환성 수준: Transact-SQL 확장

권한

자신의 것이 아닌 server_user_id에 대해 이 함수를 사용하려면 시스템 관리자 또는 시스템 보안 담당자여야 합니다.

참조

시스템 프로시저 – sp_addlogin, sp_adduser

7 장

표현식, 식별자 및 대표 문자

7장에서는 Transact-SQL 표현식, 유효한 식별자 및 대표 문자를 설명합니다.

표현식

표현식은 단일 값을 return하는 하나 이상의 상수, 리터럴, 함수, 열 식별자 및/또는 변수의 조합으로서 연산자로 구분됩니다. 표현식에는 *산술*, *관계*, *논리*(또는 *boolean*) 및 *문자열* 등 여러 유형이 있습니다. 일부 Transact-SQL 절에서는 표현식에 서브쿼리를 사용할 수 있습니다. case 식도 표현식에 사용할 수 있습니다.

표 7-1은 Adaptive Server 구문 문장에 사용되는 표현식 유형 목록입니다.

표 7-1: 구문에 사용되는 표현식 유형

사용법	정의
표현식	표현식에는 상수, 리터럴, 함수, 열 식별자, 변수 또는 매개변수 등이 포함됩니다.
논리 표현식	TRUE, FALSE, UNKNOWN을 return하는 표현식입니다.
상수 표현식	"5+3" 또는 "ABCDE" 등과 같이 항상 동일한 값을 return하는 표현식입니다.
float_expr	부동 소수점 표현식 또는 묵시적으로 부동 값으로 변환하는 표현식입니다.
integer_expr	모든 정수 표현식 또는 정수 값으로 변환하는 표현식입니다.
numeric_expr	단일 값을 return하는 모든 숫자 표현식입니다.
char_expr	단일 문자 유형 값을 return하는 모든 표현식입니다.
binary_expression	단일 binary 또는 varbinary 값을 return하는 표현식입니다.

산술 표현식과 문자 표현식

산술 표현식과 문자 표현식의 일반적인 패턴은 다음과 같습니다.

{*constant* | *column_name* | *function* | (*subquery*)
| (*case_expression*)}
[{*arithmetic_operator* | *bitwise_operator* |
string_operator | *comparison_operator* }
{*constant* | *column_name* | *function* | (*subquery*)
| *case_expression*}]...

관계 표현식 및 논리 표현식

논리 표현식 또는 관계 표현식은 TRUE, FALSE 또는 UNKNOWN을 return합니다. 일반적인 패턴은 다음과 같습니다.

expression *comparison_operator* [any | all] *expression*

expression [not] in *expression*

[not]exists *expression*

expression [not] between expression and *expression*

expression [not] like "*match_string*"
[escape *"escape_character"*]

not expression like "*match_string*"
[escape *"escape_character"*]

expression is [not] null

not *logical_expression*

logical_expression {and | or} *logical_expression*

연산자 우선 순위

연산자는 다음과 같은 우선 순위를 갖습니다. 우선 순위는 1이 가장 높고 6이 가장 낮습니다.

1 unary(단일 인자) - + ~

2 * / %

3 binary(두 인자) + - & | ^

4 not

5 and

6 or

표현식에 있는 모든 연산자가 동일한 수준일 경우 실행 순서는 왼쪽에서 오른쪽입니다. 괄호를 사용하여 실행 순서를 변경할 수 있습니다. 가장 많이 중첩된 표현식이 가장 먼저 실행됩니다.

산술 연산자

Adaptive Server는 다음 산술 연산자를 사용합니다.

표 7-2: 산술 연산자

연산자	의미
+	덧셈
–	뺄셈
*	곱셈
/	나눗셈
%	모듈로(Transact-SQL 확장)

덧셈, 뺄셈, 나눗셈, 곱셈은 정밀 숫자, 근사 숫자 및 통화 유형 열에서 사용할 수 있습니다.

모듈로 연산자는 smallmoney, money, float 또는 real 열에는 사용할 수 없습니다. 모듈로는 두 정수의 나눗셈 결과 나머지를 찾습니다. 예를 들면, 21을 11로 나누면 1이 되고 나머지는 10이므로 21 % 11 = 10이 됩니다.

float와 int처럼 혼합 데이터 유형에서 산술 연산을 할 경우 Adaptive Server는 특정한 규칙에 따라 결과 유형을 결정합니다. 자세한 내용은 1장 "시스템 및 사용자 정의 데이터 유형"을 참조하십시오.

비트 연산자

비트 연산자는 정수 유형 데이터와 함께 사용할 수 있는 Transact-SQL 확장입니다. 이 연산자는 정수 피연산자를 이진 표현으로 변환한 후 피연산자를 열 단위로 계산합니다. 1은 참이며 0은 거짓입니다.

표 7-3은 0과 1의 피연산자에 대한 결과를 요약합니다. 한 피연산자가 NULL이면 비트 연산자는 NULL을 return합니다.

표 7-3: 비트 연산의 참 테이블

&(논리곱)	1	0
1	1	0
0	0	0
|(논리합)	1	0
1	1	1
0	1	0
^(배타적 논리합)	1	0
1	0	1
0	1	0
~(부정)		
1	FALSE	
0	0	

표 7-4의 예제에서는 tinyint 인자인 A = 170 (이진 형식 10101010)과 B = 75(이진 형식 01001011)를 사용합니다.

표 7-4: 비트 연산의 예제

연산	이진 형식	결과	설명
(A & B)	10101010 01001011 ---------- 00001010	10	A와 B가 모두 1이면 결과 열은 1이며 그렇지 않으면 결과 열은 0입니다.
(A | B)	10101010 01001011 ---------- 11101011	235	A와 B 중 하나가 1이거나 둘 다 1이면 결과 열은 1이며 그렇지 않으면 결과 열은 0입니다.
(A ^ B)	10101010 01001011 ---------- 11100001	225	A와 B 중 어느 하나만 1이면 결과 열은 1입니다.
(~A)	10101010 ---------- 01010101	85	1은 모두 0으로 0은 모두 1로 바꿉니다.

문자열 연결 연산자

문자열 연산자 +는 둘 이상의 문자 또는 이진 표현식을 연결하는 데 사용할 수 있습니다. 예를 들면 다음과 같습니다.

```
select Name = (au_lname + ", " + au_fname)
from authors
```

Name 열 제목에 저자 이름을 성과 이름의 순서로 표시하며 성 다음에 콤마를 넣습니다. 예를 들면 "Bennett, Abraham"이 됩니다.

```
select "abc" + "" + "def"
```

문자열 "abc def"를 return합니다. 모든 char, varchar, unichar, nchar, nvarchar, text 연결과 varchar 및 univarchar 삽입 및 할당 구문에서 빈 문자열은 공백 한 칸으로 해석됩니다.

문자 및 이진 이외의 표현식을 연결할 때에는 항상 convert를 사용합니다.

```
select "The date is " +
    convert(varchar(12), getdate())
```

NULL로 연결된 문자열의 계산 결과는 이 문자열 값입니다. 이것은 SQL 표준의 예외로 NULL로 연결된 문자열의 계산 결과는 NULL이어야 함을 나타냅니다.

비교 연산자

Adaptive Server는 표 7-5 목록의 비교 연산자를 사용합니다.

표 7-5: 비교 연산자

연산자	의미
=	같음
>	보다 큼
<	보다 작음
>=	크거나 같음
<=	작거나 같음
<>	같지 않음
!=	같지 않음(Transact-SQL 확장)
!>	보다 크지 않음(Transact-SQL 확장)
!<	보다 작지 않음(Transact-SQL 확장)

문자 데이터 비교에서 < 는 서버 정렬 순서 시작에 더 가깝다는 것을 뜻하며 >는 정렬 순서 끝에 더 가깝다는 것을 의미합니다. 대/소문자 구분 정렬 순서에서 대문자와 소문자는 같습니다. Adaptive Server의 정렬 순서를 보려면 sp_helpsort를 사용하십시오. 비교 시에는 후미 공백을 무시합니다. 예를 들어, "Dirk"는 "Dirk"와 같습니다.

날짜를 비교할 때 <는 이전을 뜻하고 >는 이후를 뜻합니다.

비교 연산자와 함께 사용되는 모든 문자와 날짜/시간 데이터 양쪽에는 큰 따옴표나 작은 따옴표를 붙입니다.

```
= "Bennet"
> "May 22 1947"
```

비표준 연산자

다음 연산자는 Transact-SQL 확장입니다.

- 모듈로 연산자: %
- 부정 비교 연산자: !>, !<, !=
- 비트 연산자: ~, ^, |, &
- 조인 연산자: *= 및 =*

any, *all*, *in* 사용

any는 used with <, > 또는 = 및 서브쿼리와 함께 사용됩니다. 이 서브쿼리에서 검색되는 값이 외부 구문의 where 또는 having 절의 값과 일치할 경우 결과를 return합니다. 자세한 내용은 Transact-SQL User's Guide를 참조하십시오.

all은 < 또는 > 및 서브쿼리와 함께 사용됩니다. 이 연산자는 서브쿼리에서 검색되는 모든 값이 외부 구문의 where 또는 having 절의 값보다 작거나(<) 클 경우(>) 결과를 return합니다. 자세한 내용은 Transact-SQL User's Guide를 참조하십시오.

in은 두 번째 표현식의 return값이 첫 번째 표현식의 값과 일치할 경우 결과를 return합니다. 두 번째 표현식은 괄호 안에 있는 값의 목록이거나 서브쿼리여야 합니다. in은 = any와 같습니다. 자세한 내용은 where 절을 참조하십시오.

부정 및 테스트

not은 키워드나 논리 표현식의 의미를 부정합니다.

exists 다음에 서브쿼리를 사용하면 특정 결과의 존재 여부를 테스트할 수 있습니다.

범위

between은 범위 시작 키워드이며 and는 범위 끝 키워드입니다. 다음 경우에서

```
where column1 between x and y
```

는 포함됨을 의미합니다.

다음 경우에서

```
where column1 > x and column1 < y
```

는 포함되지 않음을 의미합니다.

표현식에서 null 사용

null 값을 허용하도록 정의된 열의 쿼리에 is null 또는 is not null을 사용합니다.

피연산자에 null이 있을 경우 비트 또는 산술 연산자가 있는 표현식의 계산 결과는 NULL입니다. 예를 들면 다음과 같습니다.

```
1 + column1
```

은 *column1*이 NULL일 경우 NULL이 됩니다.

TRUE를 return하는 비교

일반적으로 null 값 비교 결과는 UNKNOWN입니다. 해당 NULL이 주어진 값이나 다른 NULL과 같은지(또는 같지 않은지) 판단할 수 없기 때문입니다. 그러나 *expression*이 열, 변수 또는 리터럴이거나 이들의 조합이고 계산 결과가 NULL일 때 다음 경우에서 TRUE가 return됩니다.

- *expression* is null
- *expression* = null
- *expression* = *@x*, 여기서 *@x*는 NULL이 있는 변수이거나 매개변수입니다. 이 예외는 null 디폴트 매개변수를 가진 내장 프로시저 작성을 용이하게 합니다.
- *expression* != *n*, 여기서 *n*은 NULL이 없는 리터럴이며 *expression* 값은 NULL입니다.

이 표현식의 부정형 버전은 해당 표현식 값이 NULL이 아닌 경우 TRUE를 return합니다.

- *expression* is not null
- *expression* != null
- *expression* != *@x*

이 예외의 오른쪽 부분은 리터럴 null 또는 NULL이 있는 변수나 매개변수입니다. 이 비교의 오른쪽 부분이 표현식(예: *@nullvar* + 1)이라면 전체 표현식의 값은 NULL입니다.

이 규칙에 따르면 null 열 값은 다른 null 열 값과 조인하지 않습니다. null 열 값을 where 절의 다른 null 열 값과 비교하면 비교 연산자에 관계없이 항상 null 값에 대해 UNKNOWN이 return되며 행은 이 결과에 포함되지 않습니다. 예를 들어, 두 테이블의 column1에 모두 NULL이 있는 경우 이 쿼리는 결과 행을 return하지 않습니다. 다른 행을 return할 수도 있습니다.

```
select column1
from table1, table2
where table1.column1 = table2.column1
```

FALSE와 UNKNOWN의 차이

FALSE와 UNKNOWN은 모두 값을 return하지 않지만 거짓의 반대("not false")는 참이므로 이 둘 사이에는 중요한 논리적 차이가 있습니다. 예를 들면
"1 = 2"는 거짓이며 그 반대인 "1 != 2"는 참입니다. 하지만 "not unknown"은 아직 알 수 없는 값입니다. 비교에 null 값이 있으면 해당 표현식을 부정해도 반대 행 집합이나 반대 참 값을 얻을 수 없습니다.

"NULL" 을 문자열로 사용

create table 구문에 NULL이 지정되어 있고 NULL(따옴표 없음)을 명시적으로 입력하거나 아무 데이터도 입력하지 않은 행만 null 값을 갖습니다. 문자 행에 문자열 "NULL"(따옴표 있음)을 데이터로 입력하지 않도록 하십시오. 이것은 혼동만 줄 뿐입니다. 대신 "N/A", "none" 또는 유사한 값을 사용하십시오. 명시적으로 NULL 값을 입력하고자 할 경우에는 큰 따옴표나 작은 따옴표를 사용하지 *않아야* 합니다.

빈 문자열과 비교한 NULL

빈 문자열(" " 또는 ' ')은 변수 및 열 데이터에서 항상 공백 한 칸으로 저장됩니다. 이 연결 구문은 다음과 같습니다.

```
"abc" + "" + "def"
```

"abc def"와 동일하며 "abcdef"와는 동일하지 않습니다. 빈 문자열은 NULL과 같지 않습니다.

표현식 연결

and는 두 표현식을 연결하며 두 표현식이 모두 참일 경우 결과를 return합니다. or는 둘 이상의 조건을 연결하며 어느 한 조건이 참일 경우 결과를 return합니다.

한 구문에 논리 연산자가 두 개 이상 사용될 경우 and가 or보다 먼저 계산됩니다. 괄호를 사용하면 실행 순서를 변경할 수 있습니다.

표 7-6은 null 값이 있는 연산을 포함한 논리 연산의 결과입니다.

표 7-6: 논리 표현식의 참 테이블

and	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
NULL	UNKNOWN	FALSE	UNKNOWN
or	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
NULL	TRUE	UNKNOWN	UNKNOWN
not			
TRUE	FALSE		
FALSE	TRUE		
NULL	UNKNOWN		

UNKNOWN은 표현식 중 하나 이상이 NULL이며 따라서 연산 결과가 TRUE 또는 FALSE가 될 수 없음을 나타냅니다. 자세한 내용은 185 페이지의 "표현식에서 null 사용"을 참조하십시오.

표현식에서 괄호 사용

괄호를 사용하면 표현식의 요소를 그룹화할 수 있습니다. "표현식"이 구문 문장에서 변수로 주어지면 단순 표현식이 됩니다. "논리 표현식"은 논리 표현식이 허용되는 경우에만 지정됩니다.

문자 표현식 비교

문자 상수 표현식은 varchar로 취급됩니다. 이 표현식을 varchar 이외의 변수나 열 데이터와 비교할 경우에는 데이터 유형 우선 순위 규칙이 사용됩니다. 즉, 우선 순위가 낮은 데이터 유형은 우선 순위가 더 높은 데이터 유형으로 변환됩니다. implicit datatype conversion이 지원되지 않으면 convert 함수를 사용해야 합니다.

char 표현식과 varchar 표현식 비교는 우선 순위가 "낮은" 데이터 유형이 "높은" 데이터 유형으로 변환되는 데이터 유형 우선 순위 규칙을 따릅니다. 따라서 이 비교에서는 varchar 표현식이 모두 char 표현식으로 변환됩니다. 즉, 후미 공백이 추가됩니다. unichar 표현식을 char(varchar, nchar, nvarchar) 표현식과 비교하면 후자는 묵시적으로 unichar로 변환됩니다.

빈 문자열 사용

빈 문자열 ("") 또는 (')은 insert나 varchar 또는 univarchar 데이터에 대한 할당 구문에서 단일 공백으로 변환됩니다. varchar, char, nchar, nvarchar 데이터 연결에서 빈 문자열은 공백 한 칸으로 해석됩니다. 예를 들면 다음과 같습니다.

```
"abc" + "" + "def"
```

"abc def"로 저장됩니다. 빈 문자열은 NULL이 아닙니다.

문자 표현식에 따옴표 사용

char 또는 varchar 항목에 리터럴 따옴표를 지정하는 방법이 두 가지 있습니다. 첫 번째 방법은 따옴표를 중복 사용하는 방법입니다. 예를 들어, 작은 따옴표를 사용하여 문자 항목을 시작하는 경우 작은 따옴표를 항목의 일부로 넣으려면 다음과 같이 작은 따옴표를 두 번 사용합니다.

```
'I don''t understand. '
```

다음은 큰 따옴표를 중복 사용한 경우입니다.

```
"He said, ""It's not really confusing."""
```

두 번째 방법은 한 따옴표를 다른 종류의 따옴표에 포함시키는 방법입니다. 즉, 큰 따옴표가 있는 항목을 작은 따옴표로 둘러 싸거나 혹은 그 반대로도 가능합니다. 예를 들면 다음과 같습니다.

```
'George said, "There must be a better way."'
"Isn't there a better way?"
'George asked, "Isn"t there a better way?"'
```

연속 문자 사용

한 문자열이 화면의 다음 줄로 이어지도록 하려면 다음 줄로 가기 전에 역슬래시(\)를 입력합니다.

식별자

식별자는 데이터베이스, 테이블, 뷰, 열, 인덱스, 트리거, 프로시저, 디폴트, 규칙, 커서 등과 같은 데이터베이스 객체의 이름입니다.

Adaptive Server 식별자는 단일 바이트 문자 또는 다중 바이트 문자 사용 여부와 상관없이 최고 30바이트까지 길이가 허용됩니다. 식별자의 첫 번째 문자는 현재 문자 집합의 정의에 따라 반드시 알파벳이거나 밑줄 친(_) 문자여야 합니다.

참고 파운드 기호(#)로 시작되는 임시 테이블 이름과 @ 기호로 시작되는 로컬 변수 이름은 이 규칙에서 예외입니다.

연속 문자에는 문자, 숫자, 기호 #, @, _ 및 $(달러), ¥(엔), £(파운드)와 같은 통화 기호가 포함됩니다. 식별자에는 !, %, ^, &, *, .등의 특수 문자와 공백은 넣을 수 없습니다.

Transact-SQL 명령과 같은 예약어는 식별자로 사용할 수 없습니다. 예약어 전체 목록은 8장 "예약어"를 참조하십시오.

#로 시작하는 테이블(임시 테이블)

이름이 파운드 기호(#)로 시작하는 테이블은 임시 테이블입니다. 파운드 기호로 시작되는 다른 유형의 객체를 생성할 수 없습니다.

Adaptive Server는 임시 테이블 이름에 특별 연산을 실행하여 세션별로 고유한 이름 지정을 유지합니다. 긴 임시 테이블 이름은 13자(파운드 기호 포함)로 줄이며 짧은 이름은 밑줄(_)을 사용하여 13자로 채워집니다. Adaptive Server 세션 고유의 17자리 숫자 접미어가 추가됩니다.

대 / 소문자 구분과 식별자

식별자와 데이터의 대/소문자 구분은 Adaptive Server에 설치된 정렬 순서에 따라 달라집니다. 대/소문자 구분은 Adaptive Server의 정렬 순서를 다시 구성하여 단일 바이트 문자 집합으로 변경할 수 있습니다. 자세한 내용은 시스템 관리 지침서를 참조하십시오. 유틸리티 프로그램 옵션에서는 대/소문자 구분이 중요합니다.

Adaptive Server가 대/소문자를 구분하지 않는 정렬 순서로 설치되어 있다면 MyTable 또는 mytable이 이미 있는 경우 MYTABLE이라는 이름의 테이블을 만들 수 없습니다. 이것은 다음 명령의 경우에도 마찬가지입니다.

```
select * from MYTABLE
```

이 명령은 이름의 대/소문자에 관계없이 MYTABLE, MyTable 또는 mytable에서 행을 return합니다.

객체 이름의 고유성

데이터베이스에서 객체 이름은 고유하지 않아도 됩니다. 그러나 테이블 내의 열 이름과 인덱스 이름은 고유해야 하며 한 *데이터베이스* 내에 있는 각 *소유자* 객체 이름도 고유해야 합니다. Adaptive Server의 데이터베이스 이름도 고유해야 합니다.

구분 식별자 사용

*구분 식별자*는 따옴표에 들어 있는 객체 이름입니다. 구분 식별자를 사용하면 객체 이름에 대한 제한을 피할 수 있습니다. 따옴표로 구분할 수 있는 객체 이름은 테이블, 뷰 및 열 이름이며 다른 객체 이름은 구분할 수 없습니다.

구분 식별자는 예약어가 될 수 있으며 알파벳 이외의 문자로 시작할 수 있고 구분 식별자를 사용하지 않을 경우 허용되지 않는 문자도 사용할 수 있습니다. 28바이트를 초과할 수 없습니다.

경고! 구분 식별자는 모든 프론트엔드 응용 프로그램으로 인식할 수 있는 것은 아니며 시스템 프로시저의 매개변수로는 사용할 수 없습니다.

구분 식별자를 만들거나 참조하기 전에 먼저 다음 명령을 실행해야 합니다.

```
set quoted_identifier on
```

구문에서 구분 식별자를 사용할 때마다 큰 따옴표를 붙여야 합니다. 예를 들면 다음과 같습니다.

```
create table "1one"(col1 char(3))
create table "include spaces" (col1 int)
create table "grant"("add" int)
insert "grant"("add") values (3)
```

quoted_identifier 옵션을 사용하는 경우에는 문자 또는 날짜 문자열에 큰 따옴표 대신 작은 따옴표를 사용해야 합니다. 이 문자열을 큰 따옴표를 사용하여 구분하면 Adaptive Server가 식별자로 취급하기 때문입니다. 예를 들어, *1table*의 *col1*에 문자열을 삽입하려면 다음을 사용합니다.

```
insert "1one"(col1) values ('abc')
```

다음과 같이 사용하면 안 됩니다.

```
insert "1one"(col1) values ("abc")
```

열 내에 작은 따옴표를 삽입하려면 작은 따옴표를 연속해서 두 번 사용합니다. 예를 들어, *col1*에 "a'b"를 삽입하려면 다음과 같이 사용합니다.

```
insert "1one"(col1) values('a''b')
```

제한된 객체 이름으로 테이블과 열 식별

객체를 제한하는 다른 이름, 즉 데이터베이스 이름, 소유자 이름 및 열의 경우 테이블 또는 뷰 이름을 추가하면 테이블이나 열을 고유하게 식별할 수 있습니다. 각 식별자는 마침표를 사용하여 구분합니다. 예를 들면 다음과 같습니다.

```
database.owner.table_name.column_name
```

```
database.owner.view_name.column_name
```

이름 지정 규칙은 다음과 같습니다.

```
[[database.}owner.]table_name
```

```
[[database.}owner.]view_name
```

객체 이름에 구분 식별자 사용

set quoted_identifier on을 사용하는 경우에는 제한 객체 이름 각 부분에 큰 따옴표를 사용할 수 있습니다. 따옴표가 필요한 개별 식별자에 별도의 따옴표 쌍을 사용합니다. 예를 들면 다음과 같습니다.

```
database.owner."table_name"."column_name"
```

다음과 같이 사용하지 마십시오.

```
database.owner."table_name.column_name"
```

소유자 이름 생략

시스템에 해당 객체를 식별할 충분한 정보를 제공하는 경우에는 이름에서 중간 요소를 생략하고 점을 사용하여 그 위치를 표시할 수 있습니다.

```
database..table_name
```

```
database..view_name
```

현재 데이터베이스에서 사용자 객체 참조

데이터베이스 또는 소유자 이름을 사용하여 현재 데이터베이스에서 사용자 객체를 참조할 필요가 없습니다. *owner*의 디폴트값은 현재 사용자이며 *database*의 디폴트값은 현재 데이터베이스입니다.

데이터베이스 이름과 소유자 이름으로 제한하지 않고 객체를 참조하는 경우 Adaptive Server는 사용자가 소유한 객체 중 현재 데이터베이스에 있는 객체를 찾습니다.

데이터베이스 소유자가 소유한 객체 참조

소유자 이름을 생략했으며 해당 이름의 객체를 소유하고 있지 않은 경우 Adaptive Server는 데이터베이스 소유자의 해당 이름 객체를 찾습니다. 사용자가 같은 이름의 객체를 소유하고 있지만 데이터베이스 소유자의 객체를 사용하려는 경우에만 데이터베이스 소유자의 객체를 제한해야 합니다. 하지만 동일한 이름의 객체를 소유하고 있는지 여부에 상관없이 사용자 이름으로 다른 사용자의 객체를 제한해야 합니다.

일관된 제한 식별자 사용

동일한 구문의 열 이름과 테이블 이름을 제한할 때에는 각각 동일한 제한 표현식을 사용해야 합니다. 이것은 문자열로 평가되며 반드시 일치해야 하기 때문입니다. 그렇지 않으면 에러가 return됩니다. 다음의 두 번째 예제는 열 이름의 구문 유형이 해당 테이블 이름의 구문 유형과 일치하지 않기 때문에 잘못되었습니다.

```
1   select demo.mary.publishers.city
    from demo.mary.publishers

    city
    ----------------------
    Boston
    Washington
    Berkeley

2   select demo.mary.publishers.city
    from demo..publishers
```

열 접두어 "demo.mary.publishers"는 이 쿼리에 사용된 테이블 이름이나 가명(alias)과 일치하지 않습니다.

식별자 유효성 확인

문자 집합을 변경한 후 또는 테이블이나 뷰를 만들기 전에 시스템 함수 valid_name을 사용하여 객체 이름이 Adaptive Server에서 허용되는지 확인합니다. 구문은 다음과 같습니다.

```
select valid_name("Object_name")
```

*object_name*이 유효한 식별자가 아니면(예를 들어, 객체 이름에 유효하지 않은 문자가 들어 있거나 길이가 30바이트를 초과하는 경우) Adaptive Server는 0을 return합니다. *object_name*이 유효한 식별자이면 Adaptive Server는 0 이외의 값을 return합니다.

데이터베이스 객체 이름 변경

사용자 객체 이름(사용자 정의 데이터 유형 포함)은 sp_rename을 사용하여 변경합니다.

경고! 테이블이나 열의 이름을 변경한 후에는 이름이 변경된 개체에 의존하는 모든 프로시저, 트리거 및 뷰를 재정의해야 합니다.

다중 바이트 문자 집합 사용

다중 바이트 문자 집합에서는 식별자에 광범위한 문자를 사용할 수 있습니다. 예를 들면, 일본어가 설치된 서버에서는 젠카쿠 가타카나, 한카쿠 가타카나, 히라가나, 한자, 로마자, 그리스어, 키릴어 또는 ASCII 유형의 문자를 식별자의 첫 번째 문자로 사용할 수 있습니다.

한카쿠 가타카나 문자는 일본어 시스템에서는 식별자로 유효하지만 이기종 시스템에서는 사용하지 않는 것이 좋습니다. 이 문자는 EUC-JIS와 Shift-JIS 문자 집합 간에는 변환할 수 없습니다.

일부 8비트 유럽 문자의 경우에도 마찬가지입니다. 예를 들면, OE 합자인 "Œ" 문자는 매킨토시 문자 집합의 일부이며(codepoint 0xCE) ISO 8859-1(iso_1) 문자 집합에는 없습니다. 따라서 매킨토시에서 ISO 8859-1 문자 집합으로 변환되는 데이터에 "Œ"가 있다면 변환 에러가 발생합니다.

객체 식별자에 변환할 수 없는 문자가 들어 있는 경우 클라이언트는 해당 객체를 직접 액세스할 수 없습니다.

대표 문자를 사용한 패턴 일치

대표 문자는 *match_string*에서 하나 이상의 문자 또는 일련의 문자를 대표합니다. *match_string*은 표현식에서 찾아야 할 패턴이 들어 있는 문자열로서 다음과 같이 상수, 변수, 열 이름 또는 연결된 표현식의 모든 조합이 가능합니다.

```
like @variable + "%".
```

일치하는 문자열이 상수일 경우 항상 따옴표를 사용해야 합니다.

대표 문자와 키워드 like를 함께 사용하여 특정 패턴과 일치하는 문자 및 날짜 문자열을 찾습니다. 초 또는 밀리초 검색에는 like를 사용할 수 없습니다(201 페이지의 "대표 문자와 날짜/시간 데이터 함께 사용" 참조).

where 및 having 절에서 대표 문자를 사용하면 일치 문자열과 like 또는 not like인 문자 또는 날짜/시간 정보를 찾을 수 있습니다.

{where | having} [not]
expression [not] like *match_string*
[escape "*escape_character*"]

*expression*은 열 이름, 상수 또는 문자 값이 있는 함수의 모든 조합이 될 수 있습니다.

like 없이 사용되는 대표 문자는 특별한 의미가 없습니다. 예를 들면, 다음 쿼리는 네 문자 "415%"로 시작되는 전화 번호를 모두 찾습니다.

```
select phone
from authors
where phone = "415%"
```

not like 사용

특정 패턴과 일치하지 않는 문자열을 찾으려면 not like를 사용합니다. 다음 두 쿼리는 같습니다. 즉, 둘 다 authors 테이블에서 지역 번호가 415로 시작하지 않는 전화 번호를 모두 찾습니다.

```
select phone
from authors
where phone not like "415%"

select phone
from authors
where not phone like "415%"
```

예를 들면, 다음 쿼리는 이름이 "sys"로 시작하는 데이터베이스에서 시스템 테이블을 찾습니다.

```
select name
from sysobjects
where name like "sys%"
```

시스템 테이블이 *아닌* 객체를 모두 보려면 다음 쿼리를 사용합니다.

```
 not like "sys%"
```

모두 32개 객체를 가지고 있고 like가 패턴이 일치하는 13개 객체를 찾았다면 not like는 해당 패턴에 일치하지 않는 객체를 19개 찾게 됩니다.

not like와 부정 대표 문자[^]는 결과가 다를 수 있습니다(199 페이지의 "캐럿(^) 대표 문자" 참조). 따라서 not like 패턴을 항상 like 및 ^과 동일하게 사용할 수는 없습니다. not like는 전체 like 패턴과 일치하지 않는 항목을 찾지만 부정 대표 문자가 있는 like는 한 번에 하나의 문자로 평가되기 때문입니다.

like "[^s][^y][^s]%" 패턴은 같은 결과를 생성하지 않을 수도 있습니다. "s"로 시작하거나 *또는* 두 번째 문자가 "y"이거나 *또는* 세 번째 문자가 "s"인 이름은 시스템 테이블 이름을 포함해 모두 제거되므로 19가 아니라 14를 얻게 될 수도 있습니다. 부정 대표 문자를 사용한 일치 문자열은 한 번에 한 문자씩 단계별로 평가되기 때문입니다. 평가 도중의 어느 단계에서 일치하지 않으면 모두 제거됩니다.

대 / 소문자 및 액센트 비구분

Adaptive Server에서 대/소문자를 구분하지 않는 정렬 순서를 사용한다면 *expression*과 *match_string*을 비교할 때 대/소문자를 무시합니다. 다음 예제를 참조하십시오.

```
Where col_name like "Sm%"
```

대/소문자를 구분하지 않는 Adaptive Server에서 "Smith", "smith" 및 "SMITH"를 return합니다.

Adaptive Server가 액센트를 구분하지 않는 경우에는 대/소문자에 관계없이 강조가 있는 문자도 강조가 없는 해당 문자와 모두 동일하다고 취급합니다. sp_helpsort 시스템 프로시저는 동일하다고 취급되는 문자를 표시하며 그 사이에 "="를 넣습니다.

대표 문자 사용

많은 수의 대표 문자와 함께 일치 문자열을 사용할 수 있습니다. 이에 대해서는 다음 단원에 자세히 설명되어 있습니다. 표 7-7은 대표 문자를 요약한 것입니다.

표 7-7: like와 함께 사용되는 대표 문자

기호	의미
%	문자가 0개 이상인 문자열
_	단일 문자
[ ]	지정된 범위([a-f]) 또는 집합([abcdef])에 속하는 단일 문자
[^]	지정된 범위([^a-f]) 또는 집합([^abcdef])에 속하지 않는 단일 문자

대표 문자와 일치 문자열을 작은 따옴표나 큰 따옴표에 넣습니다(like "[dD]eFr_nce").

퍼센트 기호(%) 대표 문자

0 또는 기타 문자열을 나타낼 때에는 % 대표 문자를 사용합니다. 예를 들어, authors 테이블에서 지역 번호가 415로 시작되는 전화 번호를 찾으려면 다음과 같은 쿼리를 사용합니다.

```
select phone
from authors
where phone like "415%"
```

"en"이 들어가는 이름(예: Bennet, Green, McBadden)을 찾으려면 다음과 같습니다.

```
select au_lname
from authors
where au_lname like "%en%"
```

like 절에서 "%" 뒤의 후미 공백은 하나의 후미 공백으로 truncation 됩니다. 예를 들면, "%" 뒤에 공백이 두 개 올 경우 "X "(공백 한 개), "X "(공백 두 개), "X "(공백 세 개) 또는 후미 공백 수가 몇 개가 되더라도 모두 일치합니다.

밑줄 (_) 대표 문자

_ 대표 문자는 한 개의 문자를 나타내는 데 사용됩니다. 예를 들어 "heryl" (예: Cheryl)로 끝나는 6자 이름을 찾으려면 다음과 같이 실행합니다.

```
select au_fname
from authors
where au_fname like "_heryl"
```

대괄호 ([ ]) 문자

[a-f]처럼 일련의 문자를 포함시키거나 [a2Br]처럼 문자 집합을 포함시킬 경우 대괄호를 사용합니다. 범위를 사용하면 *rangespec1* 과 *rangespec2* 사이(포함)의 모든 값이 정렬 순서대로 return됩니다. 예를 들면, 7비트 ASCII에서 "[0-z]"는 0-9, A-Z 및 a-z(및 일부 문장 부호 문자)와 일치합니다.

"inger"로 끝나고 M과 Z 사이의 한 문자로 시작하는 이름을 찾으려면 다음과 같이 실행합니다.

```
select au_lname
from authors
where au_lname like "[M-Z]inger"
```

"DeFrance"와 "deFrance"를 모두 찾으려면 다음과 같이 실행합니다.

```
select au_lname
from authors
where au_lname like "[dD]eFrance"
```

캐럿 (^) 대표 문자

캐럿은 부정 대표 문자입니다. 이 문자를 사용하면 특정 패턴과 일치하지 않는 문자열을 찾을 수 있습니다. 예를 들면, "[^a-f]"는 a-f 범위에 속하지 않는 문자열을 찾으며 "[^a2bR]"은 "a", "2", "b" 또는 "R"이 아닌 문자열을 찾습니다.

"M"으로 시작하며 두 번째 문자가 "c"가 아닌 이름을 찾으려면 다음과 같이 실행합니다.

```
select au_lname
from authors
where au_lname like "M[^c]%"
```

범위를 사용하면 *rangespec1*과 *rangespec2* 사이(포함)의 모든 값이 정렬 순서대로 return됩니다. 예를 들면 7비트 ASCII에서 "[0-z]"는 0-9, A-Z 및 a-z(및 일부 문장 부호 문자)와 일치합니다.

다중 바이트 대표 문자 사용

Adaptive Server에 구성된 다중 바이트 문자 집합이 대표 문자 _, %, - [,], 및 ^에 해당하는 더블 바이트 문자를 정의할 경우 일치 문자열에서 해당 문자를 대신할 수 있습니다. 밑줄은 일치 문자열의 단일 또는 더블 바이트 문자를 나타냅니다.

대표 문자를 리터럴 문자로 사용

문자열 내에서 %, _, [,] 또는 ^을 검색하려면 제어 문자를 사용해야 합니다. 대표 문자를 제어 문자와 함께 사용하는 경우 Adaptive Server는 대표 문자를 다른 문자 표시에 사용하기보다는 리터럴로 해석합니다.

Adaptive Server는 두 유형의 제어 문자를 사용합니다.

- 대괄호(Transact-SQL 확장)
- escape 절 바로 뒤에 오는 하나의 문자(SQL 표준과 호환)

대괄호 ([ ]) 를 제어 문자로 사용

대괄호를 퍼센트 기호, 밑줄, 왼쪽 대괄호의 제어 문자로 사용합니다. 오른쪽 대괄호는 단독으로 사용할 수 있으므로 제어 문자가 필요하지 않습니다. 하이픈을 리터럴 문자로 사용하는 경우 대괄호 집합 내의 첫 번째 문자여야 합니다.

표 7-8은 like가 있는 제어 문자로 사용되는 대괄호의 예제입니다.

표 7-8: 대표 문자 검색에 대괄호 사용

***like* 술어**	**의미**
like "5%"	5 뒤에 0 또는 기타 문자열
like "5[%]"	5%
like "_n"	an, in, on(등)
like "[_]n"	_n
like "[a-cdf]"	a, b, c, d 또는 f
like "[-acdf]"	-, a, c, d 또는 f
like "[[]"	[
like "]"	]
like "[[]ab]"	[]ab

escape 절 사용

escape 절을 사용하여 제어 문자를 지정합니다. 서버의 디폴트 문자 집합에 속한 모든 단일 문자는 제어 문자로 사용할 수 있습니다. 둘 이상의 문자를 제어 문자로 사용하려는 경우 Adaptive Server는 예외를 생성합니다.

다음과 같은 이유로 기존의 대표 문자를 제어 문자로 사용하면 안 됩니다.

- 밑줄(_) 또는 퍼센트 기호(%)를 제어 문자로 지정하면 like 술어 내에서 특별한 의미를 잃게 되어 단지 제어 문자로서만 역할을 합니다.
- 왼쪽 또는 오른쪽 대괄호([또는])를 제어 문자로 지정하면 대괄호의 Transact-SQL 의미가 해당 like 술어 내에서 비활성화됩니다.
- 하이픈(-)이나 캐럿(^)을 제어 문자로 지정하면 해당하는 특별한 의미를 잃고 단지 제어 문자로서만 역할을 합니다.

제어 문자는 밑줄, 퍼센트 기호 및 열린 대괄호와 같은 대표 문자와는 달리 대괄호 내에서 자신의 특별한 의미를 유지합니다.

제어 문자는 해당 like 술어 내에서만 유효하며 동일한 구문 내에 포함된 다른 like 술어에는 아무런 영향을 끼치지 않습니다. 제어 문자 뒤에 사용할 수 있는 유일한 문자는 대표 문자(_, %, [,] 또는 [^])와 제어 문자입니다. 제어 문자는 바로 뒤에 오는 문자에만 영향을 끼치며 연속되는 문자는 영향을 받지 않습니다.

패턴에 제어 문자가 되기도 하는 문자의 두 개의 리터럴 발생이 포함된 경우 이 문자열은 네 개의 연속적인 제어 문자를 포함해야 합니다. 제어 문자가 패턴을 하나 또는 두 개 문자로 이루어진 조각으로 나누지 않을 경우 Adaptive Server는 에러 메시지를 return합니다. 표 7-9는 like와 함께 사용하는 escape 절의 예제입니다.

표 7-9: escape 절 사용

like 술어	의미
like "5@%" escape "@"	5%
like "*_n" escape "*"	_n
like "%80@%%" escape "@"	80%를 포함한 문자열
like "*_sql**%" escape "*"	_sql*을 포함한 문자열
like "%#####_#%%" escape "#"	##_%를 포함한 문자열

대표 문자와 날짜/시간 데이터 함께 사용

like에 날짜/시간 값을 함께 사용하면 Adaptive Server는 날짜를 표준 날짜/시간 형식으로 변환하고 다시 varchar로 변환합니다. 표준 저장 형식에는 초 또는 밀리초가 포함되지 않으므로 like 및 패턴을 사용해서는 초 또는 밀리초를 검색할 수 없습니다.

날짜/시간 항목에는 다양한 날짜 부분이 있을 수 있으므로 날짜/시간 값을 검색할 때에는 like를 사용하는 것이 좋습니다. 예를 들어, 값 "9:20"을 삽입하고 현재 날짜를 arrival_time이라는 열에 삽입하면 다음과 같은 절이 됩니다.

```
where arrival_time = '9:20'
```

Adaptive Server가 항목을 "Jan 1 1900 9:20AM"으로 변환하므로 값을 찾지 못합니다. 하지만 다음 절에서는 이 값을 찾습니다.

```
where arrival_time like '%9:20%'
```

8 장

예약어

키워드는 예약어라고도 하며 특별한 의미가 있는 단어입니다. 이 장에서는 Transact-SQL 및 SQL92 키워드 목록을 제공합니다.

Transact-SQL 예약어

다음 목록의 단어는 Adaptive Server에서 키워드(SQL 명령 구문의 일부)로 예약합니다. 이 단어는 데이터베이스, 테이블, 규칙 또는 디폴트와 같은 데이터베이스 객체 이름으로 사용할 수 없습니다. 로컬 변수 이름과 내장 프로시저 매개변수 이름으로는 사용할 수 있습니다.

예약어인 기존 객체 이름을 찾으려면 sp_checkreswords를 사용하십시오.

A

add, all, alter, and, any, arith_overflow, as, asc, at, authorization, avg

B

begin, between, break, browse, bulk, by

C

cascade, case, char_convert, check, checkpoint, close, clustered, coalesce, commit, compute, confirm, connect, constraint, continue, controlrow, convert, count, create, current, cursor

D

database, dbcc, deallocate, declare, default, delete, desc, deterministic, disk distinct, double, drop, dummy, dump

E

else, end, endtran, errlvl, errordata, errorexit, escape, except, exclusive, exec, execute, exists, exit, exp_row_size, external

F

fetch, fillfactor, for, foreign, from, func, function

G

goto, grant, group

H

having, holdlock

I

identity, identity_gap, identity_insert, identity_start, if, in, index, inout, insert, install, intersect, into, is, isolation

J

jar, join

K

key, kill

L

level, like, lineno, load, lock

M

max, max_rows_per_page, min, mirror, mirrorexit, modify

N

national, new, noholdlock, nonclustered, not, null, nullif, numeric_truncation

O

of, off, offsets, on, once, online, only, open, option, or, order, out, output, over

P

partition, perm, permanent, plan, precision, prepare, primary, print, privileges, proc, procedure, processexit, proxy_table, public

Q

quiesce

R

raiserror, read, readpast, readtext, reconfigure, references remove, reorg, replace, replication, reservepagegap, return, returns, revoke, role, rollback, rowcount, rows, rule

S

save, schema, select, set, setuser, shared, shutdown, some, statistics, stringsize, stripe, sum, syb_identity, syb_restree, syb_terminate

T

table, temp, temporary, textsize, to, tran, transaction, trigger, truncate, tsequal

U

union, unique, unpartition, update, use, user, user_option, using

V

values, varying, view

W

waitfor, when, where, while, with, work, writetext

SQL92 예약어

Adaptive Server에는 항목 수준 SQL 92 기능을 포함합니다. SQL92 전체 구현에는 다음 테이블 목록의 단어가 명령 구문으로 포함됩니다. 식별자 업그레이드는 복잡한 프로세스일 수 있으므로 사용자의 편의를 위해 이 목록을 제공합니다. 이 정보를 발표했다고 해서 Sybase가 이후의 릴리스에서 이러한 SQL92 기능을 모두 제공하는 것은 아닙니다. 또한 이후의 릴리스에는 이 목록에 없는 키워드가 포함될 수도 있습니다.

다음 목록의 단어는 Transact-SQL의 예약어가 아닌 SQL92 키워드입니다.

A

absolute, action, allocate, are, assertion

B

bit, bit_length, both

C

cascaded, case, cast, catalog, char, char_length, character, character_length, coalesce, collate, collation, column, connection, constraints, corresponding, cross, current_date, current_time, current_timestamp, current_user

D

date, day, dec, decimal, deferrable, deferred, describe, descriptor, diagnostics, disconnect, domain

E

end-exec, exception, extract

F

false, first, float, found, full

G

get, global, go

H

hour

I

immediate, indicator, initially, inner, input, insensitive, int, integer, interval

J

join

L

language, last, leading, left, local, lower

M

match, minute, module, month

N

names, natural, nchar, next, no, nullif, numeric

O

octet_length, outer, output, overlaps

P

pad, partial, position, preserve, prior

R

real, relative, restrict, right

S

scroll, second, section, session_user , size , smallint, space, sql, sqlcode, sqlerror, sqlstate, substring, system_user

T

then, time, timestamp, timezone_hour, timezone_minute, trailing, translate, translation, trim, true

U

unknown, upper, usage

V

value, varchar

W

when, whenever, write, year

Z

zone

잠재적 SQL92 예약어

ISO/IEC 9075:1989 표준을 사용하는 경우 다음 목록에 나열된 단어는 앞으로 SQL92 예약어가 될 수 있으므로 사용하지 마십시오.

A

after, alias, async

B

before, boolean, breadth

C

call, completion, cycle

D

data, depth, dictionary

E

each, elseif, equals

G

general

I

ignore

L

leave, less, limit, loop

M

modify

N

new, none

O

object, oid, old, operation, operators, others

P

parameters, pendant, preorder, private, protected

R

recursive, ref, referencing, resignal, return, returns, routine, row

S

savepoint, search, sensitive, sequence, signal, similar, sqlexception, structure

T

test, there, type

U

under

V

variable, virtual, visible

W

wait, without

9 장

SQLSTATE 코드 및 메시지

이 장에서는 Adaptive Server의 SQLSTATE 상태 코드 및 관련 메시지에 대해 설명합니다. SQLSTATE 코드는 엔트리 수준 SQL 92 호환을 위해 필요합니다. 이 코드는 두 가지 유형의 조건에 대한 진단 정보를 제공합니다.

- *경고* – 사용자 통지는 필요하지만 SQL 문을 성공적으로 실행하지 못할 정도로 심각하지는 않은 경우
- *예외* – SQL 문이 데이터베이스에 아무런 영향도 주지 못하는 경우

각 SQLSTATE 코드는 2자의 클래스와 3자의 하위 클래스로 구성됩니다. 클래스는 에러 유형에 대한 일반적인 정보를 지정합니다. 하위 클래스는 더 특정한 정보를 지정합니다.

SQLSTATE 코드는 이러한 조건이 탐지되었을 때 표시되는 메시지와 함께 sysmessages 시스템 테이블에 저장됩니다. 모든 Adaptive Server 에러 조건이 SQLSTATE 코드와 관련되어 있지는 않습니다. SQL92에서 지정한 조건만 해당합니다. 여러 Adaptive Server 에러 조건이 하나의 SQLSTATE 값과 관련되는 경우도 있습니다.

경고

Adaptive Server는 표 9-1에 설명된 하나의 SQLSTATE 경고 조건만 현재 탐지합니다.

표 9-1: SQLSTATE 경고

메시지	값	설명
경고 – set 함수에 NULL 값이 제거되었습니다.	01003	NULL 값이 포함된 식에 집계 함수(aggregate functions) (avg, max, min, sum, count)를 사용할 때 발생합니다.

예외

Adaptive Server는 다음 유형의 예외를 탐지합니다.

- 카디널리티(Cardinality) 위반
- 데이터 예외
- 무결성 제약 조건 위반
- 잘못된 커서 상태
- 구문 에러 및 액세스 규칙 위반
- 트랜잭션 롤백(rollback)
- with check option 위반

예외 조건은 표 9-2에서 표 9-8까지 설명되어 있습니다. 각 예외 클래스는 별도의 표에 표시되어 있습니다. 각 표에서 조건은 메시지 텍스트에 따라 알파벳 순서로 정렬되어 있습니다.

카디널리티(Cardinality) 위반

카디널리티(Cardinality) 위반은 Embedded SQL™ 응용 프로그램에 하나의 행만 return해야 하는 쿼리가 두 행 이상을 return할 때 발생합니다.

표 9-2: 카디널리티(Cardinality) 위반

메시지	값	설명
서브쿼리가 두 개 이상의 값을 return했습니다. 서브쿼리에 =, !=, <, <=, >, >= 등을 수반하거나 식이 사용되면 이 쿼리는 잘못되었습니다.	21000	다음의 경우에 발생합니다. • 스칼라 서브쿼리 또는 행 서브쿼리가 두 행 이상을 return한 경우 • Embedded SQL에서 select into parameter_list 쿼리가 두 행 이상을 return한 경우

데이터 예외

데이터 예외는 엔트리가 다음과 같은 경우일 때 발생합니다.

- 해당 데이터 유형에 비해 너무 긴 경우
- 잘못된 제어 시퀀스가 포함된 경우
- 다른 형식 에러가 포함된 경우

표 9-3: 데이터 예외

메시지	값	설명
산술 오버플로가 발생했습니다.	22003	다음의 경우에 발생합니다. • 정밀 숫자 유형이 산술 연산 또는 sum 함수의 결과로 정밀도(P) 또는 소수점 이하 자릿수가 손실된 경우 • 근사 숫자 유형이 truncation, 반올림 또는 sum 함수의 결과로 정밀도(P) 또는 소수점 이하 자릿수가 손실된 경우
데이터 예외 보고 – 문자열 데이터의 오른쪽이 truncation 되었습니다.	22001	char, unichar, univachar 또는 varchar 열이 삽입되거나 업데이트되는 데이터에 비해 너무 짧아 공백이 아닌 문자가 truncation될 때 발생합니다.
0(영)으로 나눔이 발생했습니다.	22012	숫자 식이 계산되고 제수의 값이 0일 때 발생합니다.
잘못된 제어 문자가 있습니다. 유효한 문자를 형성하는 데 필요한 바이트보다 작습니다.	22019	제어 시퀀스가 단일 문자로 구성되지 않은 경우 주어진 패턴과 일치하는 문자열을 검색할 때 발생합니다.
패턴 문자열이 잘못되었습니다. 제어 문자 뒤에 나오는 문자는 퍼센트 기호, 밑줄, 왼쪽 대괄호, 오른쪽 대괄호 또는 제어 문자여야 합니다.	22025	다음과 같은 특정 패턴과 일치하는 문자열을 검색할 때 발생합니다. • 제어 문자 바로 뒤에 퍼센트 기호, 밑줄 또는 제어 문자가 나오지 않는 경우 • 제어 문자가 패턴을 문자 하나 또는 둘 이외의 길이를 가진 하위 문자열로 partition하는 경우

Integrity constraint 위반

Integrity constraint 위반은 insert, update 또는 delete 문이 primary 키, foreign 키, 검사 또는 unique 제약 조건 또는 unique 인덱스를 위반할 때 발생합니다.

표 9-4: Integrity constraint 위반

메시지	값	설명
index_name unique 인덱스를 가진 *object_name* 객체에 중복키 행을 삽입하려고 했습니다.	23000	Unique 제약 조건 또는 인덱스가 있는 테이블에 중복 행이 삽입될 때 발생합니다.
check 제약 조건 위반이 발생했습니다. 데이터베이스 이름 = *database_name*, 테이블 이름 = *table_name*, 제약 조건 이름 = *constraint_name*	23000	update 또는 delete가 열에 대한 check 제약 조건을 위반할 때 발생합니다.

메시지	값	설명
참조 integrity constraint에서 종속 foreign 키 제약 조건 위반이 발생했습니다. 데이터베이스 이름 = *database_name*, 테이블 이름 = *table_name*, 제약 조건 이름 = *constraint_name*	23000	primary 키 테이블에 대한 update 또는 delete가 foreign 키 제약 조건을 위반할 때 발생합니다.
foreign 키 제약 조건 위반이 발생했습니다. 데이터베이스 이름 = *database_name*, 테이블 이름 = *table_name*, 제약 조건 이름 = *constraint_name*	23000	foreign 키에 대한 insert 또는 update가 primary 키 테이블에서 일치하는 값 없이 수행될 때 발생합니다.

잘못된 커서 상태

잘못된 커서 상태는 다음의 경우에 발생합니다.

- fetch가 현재 열리지 않은 커서를 사용하는 경우
- update where current of 또는 delete where current of가 수정되었거나 삭제된 커서 행에 영향을 미칠 경우
- update where current of 또는 delete where current of가 페치되지 않은 커서 행에 영향을 미칠 경우

표 9-5: 잘못된 커서 상태

메시지	값	설명
열려 있지 않은 *cursor_name* 커서를 열려고 합니다. 자세한 내용은 sp_cursorinfo 시스템 내장 프로시저를 참조하십시오.	24000	연 적이 없거나 commit 문 또는 묵시적 또는 명시적 rollback에 의해 닫힌 커서에서 페치를 시도할 때 발생합니다. 커서를 다시 열고 fetch를 반복하십시오.
업데이트나 삭제로 인해 현재 커서 위치가 삭제되었기 때문에 *cursor_name* 커서가 닫혔습니다. 커서 스캔 위치는 복구되지 않을 수 있습니다. 이는 하나 이상의 테이블을 참조하는 커서 때문에 발생합니다.	24000	여러 테이블 커서의 조인 열이 삭제되거나 변경되었을 때 발생합니다. 다시 fetch를 실행하여 커서 위치를 조정하십시오.
DELETE/UPDATE WHERE CURRENT OF 또는 정기적으로 검색되는 DELETE/UPDATE 때문에 *cursor_name* 커서에 삭제된 현재 스캔 위치가 있습니다. UPDATE 또는 DELETE WHERE CURRENT OF를 실행하기 전 새로운 FETCH를 실행해야 합니다.	24000	사용자가 현재 커서 위치가 삭제되거나 변경된 update/delete where current of를 실행할 때 발생합니다. update/delete where current of를 다시 시도하기 전에 다시 fetch를 실행하십시오.

메시지	값	설명
행에 있지 않기 때문에 *cursor_name* 커서에 대한 UPDATE/DELETE WHERE CURRENT OF에 오류가 발생했습니다.	24000	사용자가 다음 커서에 대해 update/delete where current of를 실행할 때 발생합니다. • 아직 행을 페치하지 않은 커서 • 결과 set의 끝까지 도착한 후 둘 이상의 행을 페치한 커서

구문 에러 및 액세스 규칙 위반

구문 에러는 종료되지 않은 주석, Adaptive Server에서 지원하지 않는 implicit datatype conversion 또는 기타 잘못된 문이 포함된 SQL 문에 의해 발생합니다.

액세스 규칙 위반은 존재하지 않는 객체 또는 사용자가 적합한 권한을 가지지 않은 객체에 대해 사용자가 액세스를 시도할 경우 발생합니다.

표 9-6: 구문 에러 및 액세스 규칙 위반

메시지	값	설명
command 사용 권한이 *object_name* 객체, *database_name* 데이터베이스, *owner_name* 소유자에서 거부되었습니다.	42000	사용자가 적합한 사용 권한을 가지지 않은 객체에 액세스를 시도할 때 발생합니다.
'*datatype*'에서 '*datatype*'으로의 데이터 유형의 implicit conversion은 허용되지 않습니다. CONVERT 함수를 사용하여 이 쿼리를 실행하십시오.	42000	사용자가 하나의 데이터 유형에서 다른 데이터 유형으로 변환을 시도했지만 Adaptive Server가 묵시적으로 변환을 수행하지 못할 때 발생합니다.
object_name 근처의 구문이 잘못되었습니다.	42000	잘못된 SQL 구문이 지정한 객체 근처에서 발견되었을 때 발생합니다.
삽입 에러: 열 이름이나 제공된 값의 숫자가 테이블 정의와 일치하지 않습니다.	42000	잘못된 열 이름이 사용되거나 잘못된 값이 삽입되었을 때 발생합니다.
주석 끝 기호(*/)를 빠뜨렸습니다.	42000	여는 주석 구분자(/*)는 있지만 닫는 주석 구분자(*/)가 없을 때 발생합니다.
*object_name*이 없습니다. owner.objectname 또는 use sp_help를 지정하여 객체가 있는지 여부를 검사하십시오(sp_help 출력 결과가 많을 수 있습니다).	42000	사용자가 소유하지 않은 객체 참조를 시도할 때 발생합니다. 다른 사용자가 소유하고 있는 객체를 참조할 경우는 해당 소유자의 이름과 함께 객체 이름을 한정하십시오.

메시지	값	설명
*object_name*에 주어진 크기(*size*)는 최대값을 초과합니다. 허용되는 최대 크기는 *size*입니다.	42000	다음의 경우에 발생합니다. • 테이블 정의 내 모든 열의 총 크기가 허용된 최대 행 크기를 초과하는 경우 • 단일 열 또는 매개변수의 크기가 해당 데이터 유형에 허용된 최대 크기를 초과하는 경우

트랜잭션 롤백(rollback)

트랜잭션 롤백(rollback)은 트랜잭션 isolation level이 3으로 설정되어 있지만, Adaptive Server가 동시 트랜잭션의 직렬화를 보장할 수 없을 때 발생합니다. 이런 유형의 예외는 일반적으로 디스크 충돌 및 오프라인 디스크와 같은 시스템 문제로 인해 발생합니다.

표 9-7: 트랜잭션 롤백(rollback)

메시지	값	설명
서버 명령(프로세스 ID #*process_id*)이 다른 프로세스와 데드락(deadlock)이 되었으며 데드락(deadlock) 빅팀으로 선택되었습니다. 명령을 다시 실행하십시오.	40001	Adaptive Server가 둘 이상의 동시 트랜잭션을 직렬화할 수 없음을 탐지할 때 발생합니다.

with check option 위반

이 클래스의 예외는 뷰를 통해 삽입되거나 업데이트되는 데이터를 뷰를 통해 볼 수 없는 경우 발생합니다.

표 9-8: with check option 위반1

메시지	값	설명
대상 뷰가 WITH CHECK OPTION으로 만들어졌거나 WITH CHECK OPTION을 만든 다른 뷰에 걸쳐 있기 때문에 삽입하거나 업데이트할 수 없습니다. 이 명령을 실행한 결과 행이 CHECK OPTION 제약 조건으로 한정되지 않았습니다.	44000	뷰 또는 뷰가 종속하는 뷰가 확인 옵션 절로 만들어졌을 때 발생합니다.

색인

기호

숫자

가

나

다

라

마

바

사

아

자

차

카

타

파

하

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Y