데이터베이스와 SQL

7장

데이터 생성, 조작과 변환

빅데이터 분석가 과정

목차

7.1 문자열 데이터 처리

- char
  - 고정 길이 문자열 자료형
  - 지정한 크기보다 문자열이 작으면 나머지 공간을 공백으로 채움
  - MySQL: 255글자

- varchar
  - 가변 길이 문자열 자료형
  - 크기만큼 데이터가 들어오지 않으면 그 크기에 맞춰 공간 할당
  - 헤더에 길이 정보가 포함
  - MySQL 최대 65,536 글자 허용

- text
  - 매우 큰 가변 길이 문자열 저장
  - MySQL: 최대 4 기가바이트 크기 문서 저장
  - clob: 오라클 데이터베이스

7.1 문자열 데이터 처리

- 테이블 생성

```
use sqlclass_db;

DROP TABLE IF EXISTS string_tbl;
CREATE TABLE string_tbl
(char_fld CHAR(30),
vchar_fld VARCHAR(30),
text_fld TEXT
);
```

- 문자열 데이터를 테이블에 추가
  - 문자열의 길이가 해당 열의 최대 크기를 초과하면 예외 발생

```
INSERT INTO string_tbl (char_fld, vchar_fld, text_fld)
VALUES ('This is char data',
        'This is varchar data',
        'This is text data');
```

7.1 문자열 데이터 처리

- 저장된 내용 확인

```
SELECT * FROM string_tbl;
```

```
char_fld         |vchar_fld           |text_fld         |
-----------------+--------------------+-----------------+
This is char data|This is varchar data|This is text data|
```

- varchar 문자열 처리
  - update문으로 vchar_fld열 (varchar(30))에 설정 길이보다 더 긴 문자열 저장
  - MySQL 6.0 이전 버전: 문자열을 최대 크기로 자르고 경고 발생
  - MySQL 6.0 이후 기본 모드는 strict 모드로 예외 발생됨

```
UPDATE string_tbl
SET vchar_fld = 'This is a piece of extremely long varchar data';
```

```
SQL Error [1406] [22001]: Data truncation: Data too long for column 'vchar_fld' at row 1
```

7.1 문자열 데이터 처리

- ANSI 모드 선택
  - 문자열을 최대 크기로 자르고 경고만 생성
  - 현재 모드 확인

```
SELECT @@session.sql_mode;
```

```
@@session.sql_mode                                                                                     ¦
-------------------------------------------------------------------------------------------------------+
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION¦
```

  - ANSI 모드 선택

```
SET sql_mode='ansi';
```

    - 변경 사항 확인

```
SELECT @@session.sql_mode;
```

```
@@session.sql_mode                                                      ¦
------------------------------------------------------------------------+
REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ONLY_FULL_GROUP_BY,ANSI¦
```

    - vchar_fld 업데이트 실행: ANSI 모드로 변경 후 문자열이 잘려서 저장됨

```
UPDATE string_tbl
SET vchar_fld = 'This is a piece of extremely long varchar data';
select vchar_fld from string_tbl;
```

```
vchar_fld                      ¦
-------------------------------+
This is a piece of extremely l¦
```

7.1 문자열 데이터 처리

- 작은 따옴표 포함
  - 문자열 내부에 작은 따옴표를 포함하는 경우 (I'm, doesn't 등)
  - escape 문자 추가 방법
    - 작은 따옴표를 하나 더 추가

```
update string_tbl
set text_fld = 'This string didn''t work, but it does now';
```

    - 백슬래시('\') 문자 추가

```
update string_tbl
set text_fld = 'This string didn\'t work, but it does now';
```

```
select text_fld from string_tbl;
```

```
text_fld                                  ¦
------------------------------------------+
This string didn't work, but it does now¦
```

7.1 문자열 데이터 처리

- 작은 따옴표 포함
  - quote() 내장 함수
  - 전체 문자열을 따옴표로 묶고, 문자열 내부의 작은 따옴표에 escape문자를 추가

```
select quote(text_fld)
from string_tbl;
```

```
quote(text_fld)                                 ¦
------------------------------------------------+
'This string didn\'t work, "but it does now."'¦
```

7.1.2 문자열 조작

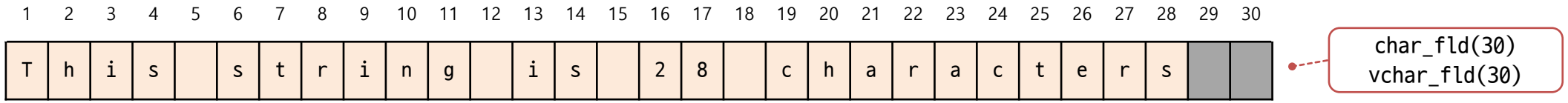
- length() 함수: 문자열의 개수를 반환

```
# string_tbl 내용 삭제
DELETE FROM string_tbl;

INSERT INTO string_tbl(char_fld, vchar_fld, text_fld)
VALUES ('This string is 28 characters',
        'This string is 28 characters',
        'This string is 28 characters');
```

```
SELECT length(char_fld) as char_length,
length(vchar_fld) as varchar_length,
length(text_fld) as text_length
FROM string_tbl;
```

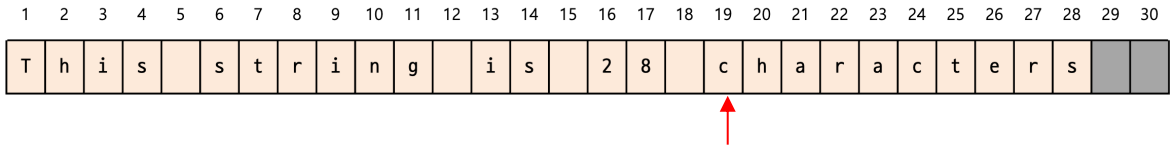
```
char_length¦varchar_length¦text_length¦
-----------+--------------+-----------+
         28¦            28¦         28¦
```

- char열의 길이: 빈 공간을 공백으로 채우지만, 조회할 때 char데이터에서 공백 제거



7.1.2 문자열 조작

- position() 함수
  - 부분 문자열의 위치를 반환
  - MySQL의 문자열 인덱스: 1부터 시작
  - 부분 문자열을 찾을 수 없는 경우, 0을 반환함




```
SELECT position('characters' in vchar_fld)
FROM string_tbl;
```

```
position('characters' in vchar_fld)¦
------------------------------------+
                                  19¦
```

- locate('문자열', 열이름, 시작위치) 함수
  - 시작 위치부터 문자열 검색: 처음 발견되는 인덱스 리턴

```
SELECT locate('is', vchar_fld, 5)
FROM string_tbl;
```

```
locate('is', vchar_fld, 5)¦
--------------------------+
                        13¦
```

```
SELECT locate('is', vchar_fld, 1)
FROM string_tbl;
```

```
LOCATE('is', vchar_fld, 1)¦
--------------------------+
                        3¦
```

7.1.2 문자열 조작

- strcmp('문자열1', '문자열2') 함수: 문자열 비교
  - 대소문자 구분 안함
    - if 문자열1 < 문자열2, -1 반환: 정렬 순서에서 문자열1이 문자열2의 앞에 오는 경우
    - if 문자열1 == 문자열2, 0 반환: 문자열이 동일
    - if 문자열1 > 문자열2, 1 반환: 문자열1이 문자열2의 뒤에 오는 경우

- string_tbl 삭제 후 새로운 데이터 추가

```
DELETE FROM string_tbl;

INSERT INTO string_tbl(vchar_fld)
VALUES ('abcd'),
       ('xyz'),
       ('QRSTUV'),
       ('qrstuv'),
       ('12345');
```

- vchar_fld의 값을 오름 차순 정렬

```
select vchar_fld from string_tbl order by vchar_fld;
```

```
vchar_fld
---------
12345
abcd
QRSTUV
qrstuv
xyz
```

7.1.2 문자열 조작

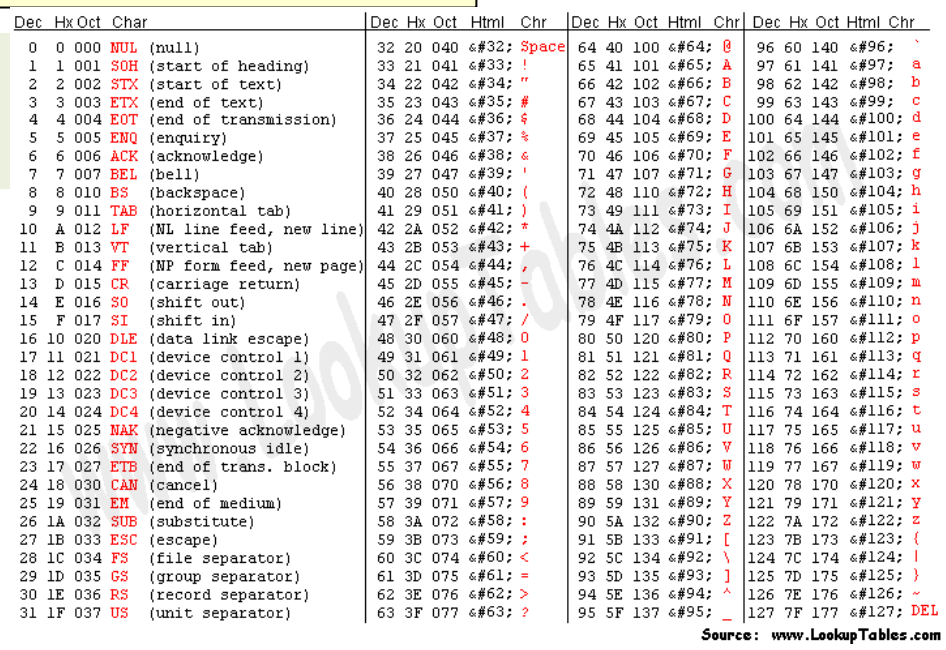
- strcmp() 예제
  - 5개의 서로 다른 문자열 비교

```
SELECT strcmp('12345', '12345') 12345_12345,
       strcmp('abcd', 'xyz') abcd_xyz,
       strcmp('abcd', 'QRSTUV') abcd_QRSTUV,
       strcmp('qrstuv', 'QRSTUV') qrstuv_QRSTUV,
       strcmp('12345', 'xyz') 12345_xyz,
       strcmp('xyz', 'qrstuv') xyz_qrstuv;
```

대소문자 구분 안함

```
12345_12345¦abcd_xyz¦abcd_QRSTUV¦qrstuv_QRSTUV¦12345_xyz¦xyz_qrstuv¦
------------+--------+-----------+-------------+---------+----------+
           0¦      -1¦         -1¦            0¦       -1¦         1¦
```

- 대소문자 구분 안함



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

ASCII Table

7.1.2 문자열 조작(문자열 비교)

- like 또는 regexp 연산자 사용
  - select 절에 like 연산자나 regexp 연산자를 사용
    - 0 또는 1의 값을 반환

```
use sakila;

SELECT name, name LIKE '%y' as ends_in_y
FROM category;

SELECT name, name REGEXP 'y$' as ends_in_y
FROM category;
```

- 'y$': name 컬럼의 값이 'y'로 끝나면 1을 반환

```
name       |ends_in_y|
-----------+---------+
Action     |        0|
Animation  |        0|
Children   |        0|
Classics   |        0|
Comedy     |        1|
Documentary|        1|
Drama      |        0|
Family     |        1|
Foreign    |        0|
Games      |        0|
Horror     |        0|
Music      |        0|
New        |        0|
Sci-Fi     |        0|
Sports     |        0|
Travel     |        0|
```

7.1.2 문자열 조작

- string_tbl 리셋

```
use sqlclass_db;
DELETE FROM string_tbl;

INSERT INTO string_tbl (text_fld)
VALUES ('This string was 29 characters');
```

- concat(): 문자열 추가 함수
  - concat() 함수를 사용하여 string_tbl의 text_fld열에 저장된 문자열 수정
    - 기존 text_fld의 문자열에 ', but now it is longer' 문자열 추가

```
UPDATE string_tbl
SET text_fld = concat(text_fld, ', but now it is longer');
```

추가

  - 변경된 text_fld 열 확인

```
SELECT text_fld FROM string_tbl;
```

```
text_fld                                          ¦
--------------------------------------------------+
This string was 29 characters, but now it is longer¦
```

7.1.2 문자열 조작

- concat() 함수 활용
  - 각 데이터 조각을 합쳐서 하나의 문자열 생성
    - concat() 함수 내부에서 date(create_date)를 문자열로 변환

```
use sakila;
# concat() 함수 사용 #2
SELECT concat(first_name, ' ', last_name,
' has been a customer since ', date(create_date)) as cust_narrative
FROM customer;
```

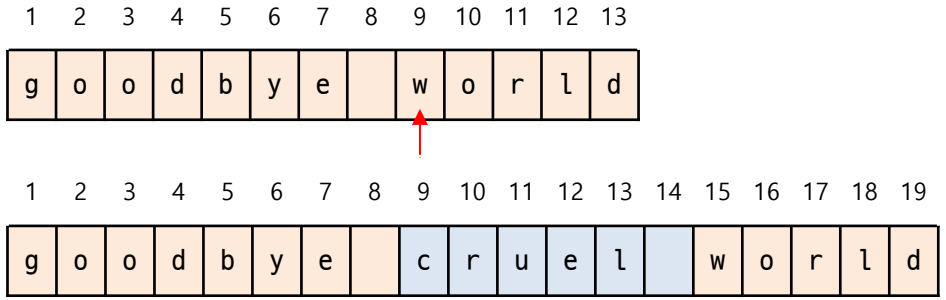
```
cust_narrative                                          ¦
--------------------------------------------------------+
MARY SMITH has been a customer since 2006-02-14         ¦
PATRICIA JOHNSON has been a customer since 2006-02-14   ¦
LINDA WILLIAMS has been a customer since 2006-02-14     ¦
BARBARA JONES has been a customer since 2006-02-14      ¦
ELIZABETH BROWN has been a customer since 2006-02-14    ¦
JENNIFER DAVIS has been a customer since 2006-02-14     ¦
MARIA MILLER has been a customer since 2006-02-14       ¦
SUSAN WILSON has been a customer since 2006-02-14       ¦
MARGARET MOORE has been a customer since 2006-02-14     ¦

. . .
```

7.1.2 문자열 조작

- insert() 함수
  - 4개의 인수로 구성
  - insert(문자열, 시작위치, 길이, 새로운 문자열)
    - 세 번째 인수값(길이)=0: 새로운 문자열이 삽입

```
SELECT INSERT('goodbye world', 9, 0, 'cruel ') as string;
```

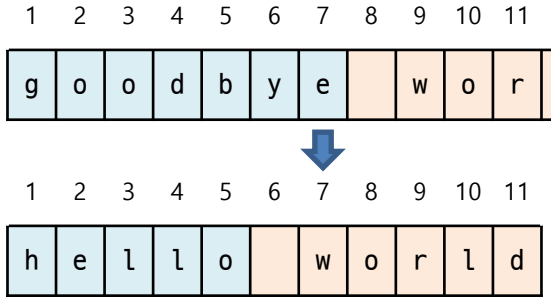
```
string              ¦
--------------------+
goodbye cruel world¦
```



    - 세 번째 인수값 > 0: 해당 문자열로 대치

```
SELECT INSERT('goodbye world',1, 7, 'hello') as string;
```

```
string     ¦
-----------+
hello world¦
```



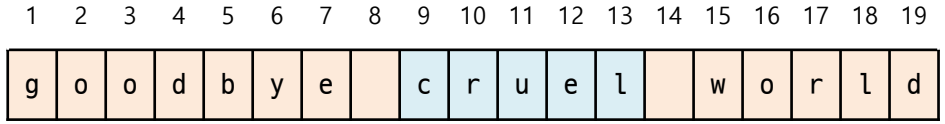
7.1.2 문자열 조작

- replace() 함수
  - replace(문자열, 기존문자열, 새로운 문자열)
  - 기존 문자열을 찾아서 새로운 문자열로 교체

```
SELECT replace('goodbye world', 'goodbye', 'hello') as replace_str;
```

```
replace_str|
-----------+
hello world|
```

- substr() 또는 substring() 함수
  - substr(문자열, 시작위치, 개수)
  - 문자열에서 시작 위치에서 개수만큼 추출




```
SELECT substr('goodbye cruel world', 9, 5);
```

```
substr('goodbye cruel world', 9, 5)|
-----------------------------------+
cruel                              |
```

7.2 숫자 데이터 처리

■ 산술 함수

산술함수	설명
cos(x)	x의 코사인 계산
cot(x)	x의 코탄젠트 계산
ln(x)	x의 자연로그 계산
sin(x)	x의 사인 계산
sqrt(x)	x의 제곱근 계산
tan(x)	x의 탄젠트 계산
exp(x)	e^x 를 계산
mod(a, b)	a를 b로 나눈 나머지 구하기
pow(a, b)	a의 b 제곱근 계산
sign(x)	x가 음수이면 -1, 0이면 0, 양수이면 1 반환
abs(x)	x의 절대값 계산

7.2 숫자 데이터 처리

- 숫자 자릿수 관리
  - ceil() 함수: 가장 가까운 정수로 올림
    - ceil(72.445) = 73
  - floor() 함수: 가장 가까운 정수로 내림
    - floor(72.445) = 72
  - round() 함수: 반올림
    - 소수점 자리를 정할 수 있음
    - round(72.0909, 1) = 72.1
    - round(72.0909, 2) = 72.09

```
select round(72.0909, 1), round(72.0909, 2), round(72.0909, 3);
```

```
round(72.0909, 1)¦round(72.0909, 2)¦round(72.0909, 3)¦
-----------------+-----------------+-----------------+
            72.1¦            72.09¦           72.091¦
```

  - truncate(숫자, 자릿수) 함수: 자릿수 아래를 버림
    - truncate(72.0956, 1) = 72.0
    - truncate(72.0956, 2) = 72.09
    - truncate(72.0956, 3) = 72.095

```
select truncate(72.0956, 1), truncate(72.0956, 2), truncate(72.0956, 3);
```

```
truncate(72.0956, 1)¦truncate(72.0956, 2)¦truncate(72.0956, 3)¦
--------------------+--------------------+--------------------+
                72.0¦               72.09¦              72.095¦
```

7.2 숫자 데이터 처리

- sign() 함수
  - 값이 음수이면 -1, 0이면 0, 양수이면 1을 반환

```
use sqlclass_db;
DROP TABLE IF EXISTS account;
# account 테이블 생성 (7장 p.188)
CREATE TABLE account
    (account_id int,
    acct_type varchar(20),
    balance float);

INSERT INTO account (account_id, acct_type, balance)
VALUES (123, 'MONEY MARKET', 785),
       (456, 'SAVINGS', 0.00),
       (789, 'CHECKING', -324);
```

```
+---------+------------+-----------+
|account_id|   acct_type |    balance |
+---------+------------+-----------+
|       123| MONEY_MARKET|      785.22|
|       456| SAVINGS     |         0.0|
|       789| CHECKING    |     -324.22|
+---------+------------+-----------+
```

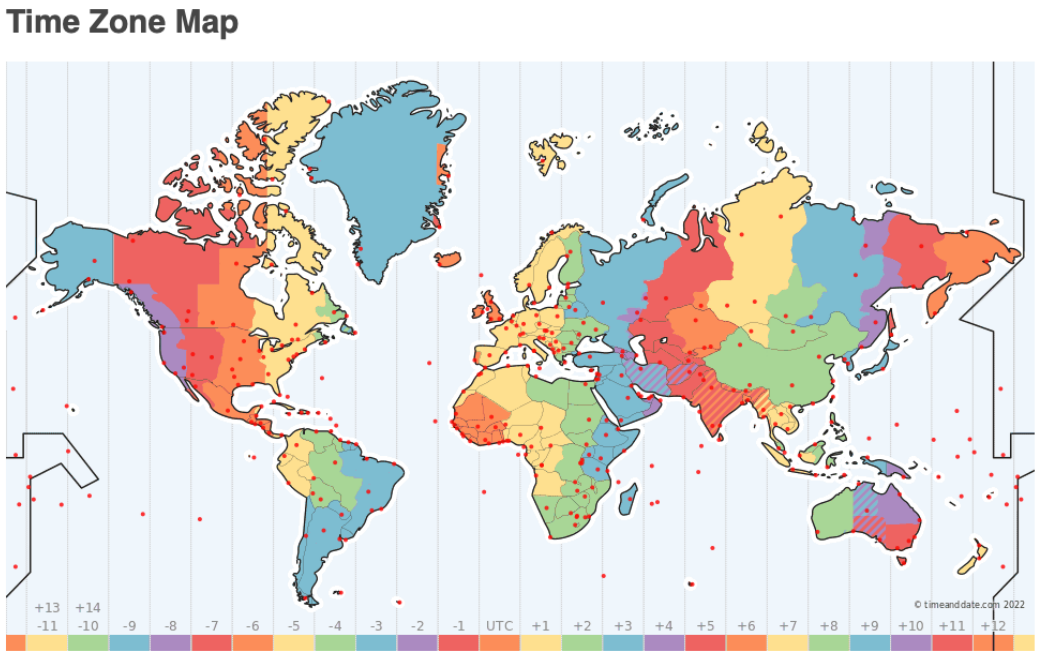
```
SELECT account_id, sign(balance), abs(balance) FROM account;
```

```
account_id|SIGN(balance)|ABS(balance)|
----------+-------------+------------+
       123|            1|       785.0|
       456|            0|         0.0|
       789|           -1|       324.0|
```

7.3 시간 데이터 처리

- 시간대(time zone)처리
  - 24개의 가상 영역으로 분할
  - 협정 세계표준시(UTC: Universal Time Coordinated) 사용
  - utc_timestamp() 함수 제공

- 시간 데이터 생성 방법
  - 기존 date, datetime 또는 time 열에서 데이터 복사
  - date, datetime 또는 time을 반환하는 내장 함수 실행
  - 서버에서 확인된 시간 데이터를 문자열로 표현



https://www.timeanddate.com/time/map/

7.3 시간 데이터 처리

■ 날짜 형식의 구성 요소

자료형	기본 형식	허용값
YYYY	연도	1000 ~ 9999
MM	월	01(1월) ~ 12(12월)
DD	일	01 ~ 31
HH	시간	00 ~ 23
MI	분	00 ~ 59
SS	초	00 ~ 59

■ 필수 날짜 구성 요소

자료형	기본 형식	허용값
date	YYYY-MM-DD	1000-01-01 ~ 9999-12-31
datetime	YYYY-MM-DD HH:MI:SS	1000-01-01 00:00:00.000000 ~ 9999-12-31 23:59:59.999999
timestamp	YYYY-MM-DD HH:MI:SS	1970-01-01 00:00:00.000000 ~ 2038-01-18 22:14:07.999999
time	HHH:MI:SS	-838:59:59.000000 ~ 838:59:59.000000

7.3 시간 데이터 처리

- 시간 데이터의 문자열 표시
  - datetime 기본 형식: YYYY-MM-DD HH:MI:SS
  - datetime 열을 2022년 8월 1일 오전 09:30 으로 표현
    - '2022-08-01 09:30:00' 의 문자열로 구성

  - MySQL 서버의 시간 데이터 처리
    - datetime 형식으로 표현된 문자열에서 6개의 구성요소를 분리해서 문자열을 변환

  - cast() 함수
    - 지정한 값을 다른 데이터 타입으로 변환
    - cast() 함수를 이용해서 datetime값을 반환하는 쿼리 생성

```
SELECT cast('2019-09-17 15:30:00' as datetime);
```

```
cast('2019-09-17 15:30:00' as datetime)¦
---------------------------------------+
                   2019-09-17 15:30:00¦
```

7.3 시간 데이터 처리

- cast() 함수
  - date 값과 time 값을 생성

```
SELECT cast('2019-09-17' as date) date_field,
       cast('108:17:57' as time) time_field;
```

```
date_field|time_field|
----------+----------+
2019-09-17| 108:17:57|
```

- MySQL의 문자열을 이용한 datetime 처리
  - MySQL은 날짜 구분 기호에 관대
    - 2019년 9월 17일 오후 3시 30분에 대한 유효한 표현 방식

```
'2019-09-17 15:30:00'
'2019/09/17 15:30:00'
'2019,09,17,15,30,00'
'20190917153000'
```

```
SELECT cast('20190917153000' as datetime);
```

```
cast('20190917153000' as datetime)|
----------------------------------+
                2019-09-17 15:30:00|
```

7.3 시간 데이터 처리

- 날짜 생성 함수
  - str_to_date(str, format)
    - 형식 문자열의 내용에 따라 datetime, date 또는 time값을 반환
    - cast() 함수를 사용하기에 적절한 형식이 아닌 경우 사용
    - 'September 17, 2019' 문자열을 date 형식으로 변환
      - str(문자열)의 형식에 맞춰 format을 설정

```
SELECT str_to_date('September 17, 2019', '%M %d, %Y') as return_date;
```

```
return_date|
-----------+
 2019-09-17|
```

      - %M: 월 이름 (January ~ December)
      - %d: 숫자로 나타낸 월(01 ~ 12)
      - %Y: 연도, 4자리 숫자

7.3 시간 데이터 처리

- 날짜 형식의 구성 요소: format

요소	정의	요소	정의
%M	월 이름 (January ~ December)	%H	시간 (00 ~ 23)
%m	숫자로 나타낸 월 (01 ~12)	%h	시간 (01 ~ 12)
%d	숫자로 나타낸 일 (01 ~ 31)	%i	분 (00 ~ 59)
%j	일년 중 몇 번째 날 (001 ~ 366)	%s	초 (00 ~ 59)
%W	요일 이름 (Sunday ~ Saturday)	%f	마이크로초 (000000 ~ 999999)
%Y	연도, 4자리 숫자	%p	오전 또는 오후
%y	연도, 2자리 숫자		

7.3 시간 데이터 처리

- str_to_date(str, format) 예제
  - 날짜 정보가 슬래쉬('/')로 구분되어 있음

```
SELECT str_to_date('04/31/2024', '%m/%d/%Y') as date1;
```

```
date1      ¦
-----------+
2024-04-31¦
```

  - 일, 월, 연도로 표시된 문자열을 날짜로 변환

```
SELECT str_to_date('01,5,2024', '%d,%m,%Y') as date2;
```

```
date2      ¦
-----------+
2024-05-01¦
```

  - 시간 문자열을 time값으로 변환

```
SELECT str_to_date('15:35:00', '%H:%i:%s') as time1;
```

```
time1    ¦
---------+
15:35:00¦
```

7.3 시간 데이터 처리

- 현재 날짜/시간 생성
  - 내장 함수가 시스템 시계를 확인해서 현재 날짜 및 시간을 문자열로 반환
  - CURRENT_DATE(), CURRENT_TIME(), CURRENT_TIMESTAMP()

```
SELECT CURRENT_DATE(), CURRENT_TIME(), CURRENT_TIMESTAMP();
```

```
CURRENT_DATE()¦CURRENT_TIME()¦CURRENT_TIMESTAMP()¦
--------------+--------------+--------------------+
    2022-06-30¦      19:57:48¦2022-06-30 19:57:48¦
```

- 날짜를 반환하는 시간 함수
  - date_add()
    - 지정한 날짜에 일정 기간(일, 월, 년 등)을 더해서 다른 날짜를 생성

```
SELECT date_add(current_date(), interval 5 day);
```

```
date_add(current_date(), interval 5 day)¦
-----------------------------------------+
                              2022-07-05¦
```

7.3 시간 데이터 처리

- 기간 자료형

기간명	정의
second	초
minute	분
hour	시간
day	일 수
month	개월 수
year	년 수
minute_second	':'으로 구분된 분, 초
hour_second	':'으로 구분된 시, 분, 초
year_month	'-' 으로 구분된 년, 월

7.3 시간 데이터 처리

- 날짜를 반환하는 시간 함수
  - last_day(date)
    - 해당월의 마지막 날짜 반환

```
SELECT last_day('2022-08-01');
```

```
last_day('2022-08-01')¦
----------------------+
           2022-08-31¦
```

- 문자열을 반환하는 시간 함수
  - dayname(date) 함수
    - 해당 날짜의 영어 요일 이름을 반환

```
SELECT dayname('2022-08-01');
```

```
dayname('2022-08-01')¦
---------------------+
Monday               ¦
```

7.3 시간 데이터 처리

- 문자열을 반환하는 시간 함수
  - extract() 함수
    - date의 구성 요소 중 일부를 추출
    - 기간 자료형으로 원하는 날짜 요소를 정의

```
SELECT extract(year FROM '2019-09-18 22:19:05');
```

```
extract(year from '2019-09-18 22:19:05')¦
----------------------------------------+
                                   2019¦
```

- 숫자를 반환하는 시간 함수
  - datediff(date1, date2) 함수
    - 두 날짜 사이의 기간(년, 주, 일)을 계산
    - 시간 정보는 무시

```
SELECT datediff('2019-09-03', '2019-06-21');
```

```
datediff('2019-09-03', '2019-06-21')¦
-------------------------------------+
                                  74¦
```

7.4 변환 함수

- 변환 함수
  - cast() 함수
    - 데이터를 한 유형에서 다른 유형으로 변환할 때 사용
    - cast(데이터 as 타입)

```
SELECT cast('1456328' as signed integer);
```

```
cast('1456328' as signed integer)¦
---------------------------------+
                         1456328¦
```

    - cast() 예제

```
SELECT cast('20220101' as date);
```

```
cast('20220101' as date)¦
-------------------------+
              2022-01-01¦
```

```
SELECT cast(now() as signed);
```

```
cast(now() as signed)¦
----------------------+
       20240805201543¦
```

Questions?