제10장

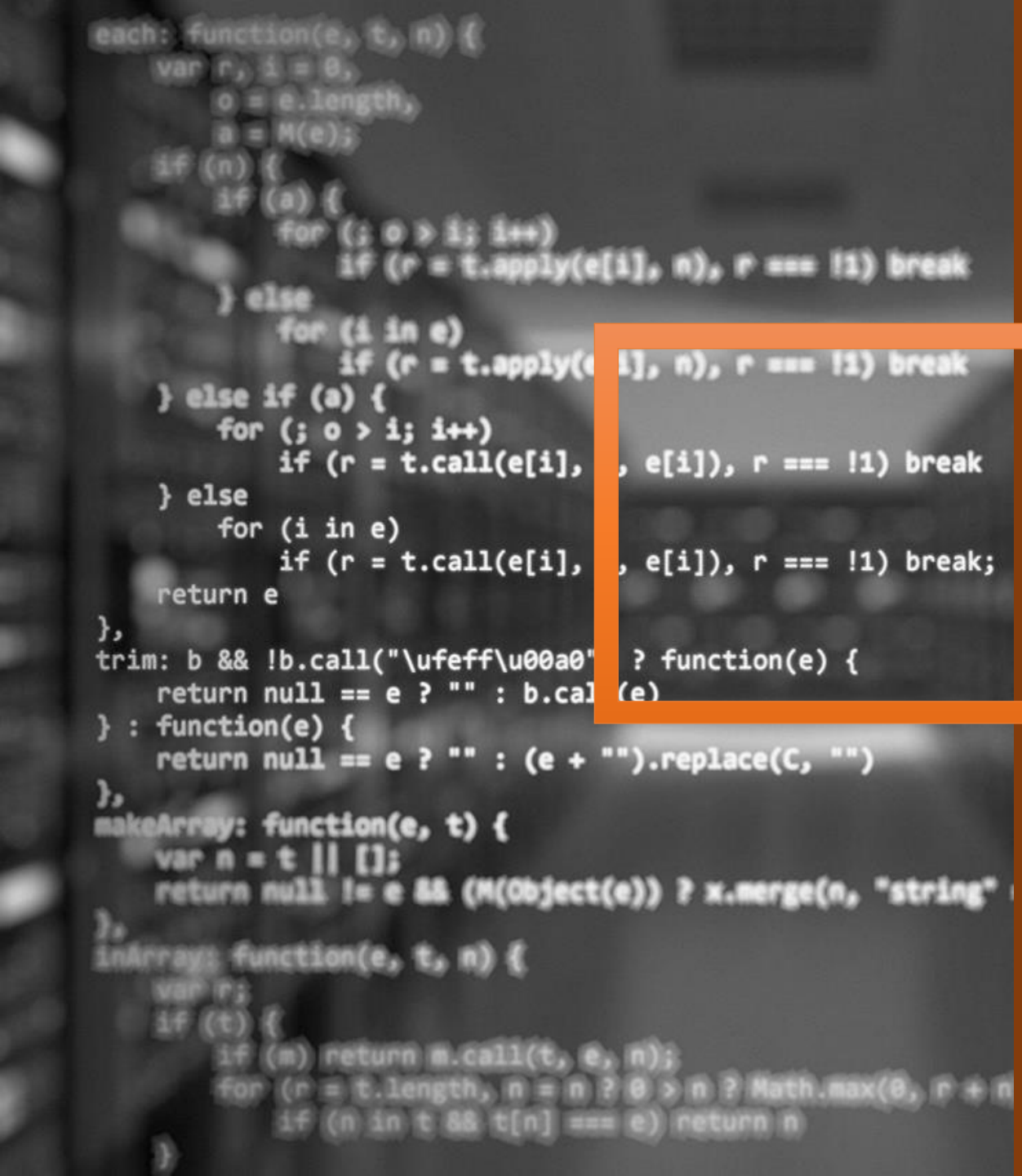
내장 함수

MySQL

학습목표

1. 문자열 함수에 알 수 있다.
2. 수학 함수에 대해서 알 수 있다.
3. 날짜/시간 함수에 대해서 알 수 있다.
4. 기타 유용한 함수에 대해서 알 수 있다.

목차



1. 문자열 함수

ASCII

- ASCII
  - 전달된 문자의 아스키 코드 값을 반환
- 형식
  - ASCII(str)
- 실행
  - 쿼리 : `SELECT ASCII('A'), ASCII('a');`
  - 결과 : 65, 97

CHAR

- CHAR
  - 전달된 아스키 코드 값을 가진 문자를 반환

- 형식
  - CHAR(n)

- 실행
  - 쿼리 : SELECT CHAR(65), CHAR(97);
  - 결과 : 'A', 'a'
    - Workbench 에서는 결과가 BLOB으로 보일 수 있는데 일반 명령형 모드에서는 정상적으로 출력된다. BLOB에서 우클릭하고 'Open Value in Viewer' 선택한 뒤 Text 탭에서 결과를 확인할 수 있다.

CHAR_LENGTH

- CHAR_LENGTH
  - 전달된 문자열의 길이를 반환

- 형식
  - CHAR_LENGTH(str)

- 실행
  - 쿼리 : `SELECT CHAR_LENGTH('hello'), CHAR_LENGTH('안녕하세요');`
  - 결과 : 5, 5

LENGTH

- LENGTH
  - 전달된 문자열의 BYTE 크기를 반환

- 형식
  - LENGTH(str)

- 실행
  - 쿼리 : `SELECT LENGTH('hello'), LENGTH('안녕하세요');`
  - 결과 : 5, 15
    - UTF-8 인코딩을 하게 되면 영문은 한 글자당 1바이트, 한글은 한 글자당 3바이트의 크기를 가진다.

CONCAT

- CONCAT
  - 전달된 문자열을 모두 이어 붙인 결과를 반환
- 형식
  - CONCAT(str1, str2, str3, ...)
- 실행
  - 쿼리 : SELECT CONCAT('아이스', '크림', '라떼');
  - 결과 : '아이스크림라떼'

CONCAT_WS

- CONCAT_WS
  - 구분자와 함께 전달된 문자열을 모두 이어 붙인 결과를 반환
- 형식
  - CONCAT_WS(separator, str1, str2, str3, ...)
- 실행
  - 쿼리 : SELECT CONCAT_WS('-', '아이스', '크림', '라떼');
  - 결과 : '아이스-크림-라떼'

ELT

- ELT
  - 지정한 n번째 문자열을 반환

- 형식
  - ELT(n, str1, str2, str3, ...)

- 실행
  - 쿼리 : SELECT ELT(2, '돼지', '국밥', '만세'), ELT(5, '돼지', '국밥', '만세');
  - 결과 : '국밥', NULL
    - n번째 문자열이 없으면 NULL을 반환한다.

FIELD

- FIELD
  - 지정한 문자열의 위치를 반환
- 형식
  - FIELD(str, str1, str2, str3, ...)
- 실행
  - 쿼리 : SELECT FIELD('국밥', '돼지', '국밥', '만세'), FIELD('순두부', '돼지', '국밥', '만세');
  - 결과 : 2, 0
    - 지정한 문자열이 없으면 0을 반환한다.

FORMAT

- FORMAT
  - 전달된 값(X)과 소수 자리수(D)를 이용해 반올림한 결과를 '#,###,###.##' 형식으로 반환함

- 형식
  - FORMAT(X, D)

- 실행
  - 쿼리 : SELECT FORMAT(12332.123456, 4), FORMAT(12332.1,4), FORMAT(12332.2,0);
  - 결과 : '12,332.1235', '12,332.1000', '12,332'

BIN / OCT / HEX

- BIN / OCT / HEX
  - 전달된 값(X)의 2진수(BIN), 8진수(OCT), 16진수(HEX) 문자열을 반환
- 형식
  - BIN(N), OCT(N), HEX(N) / HEX(str)
- 실행
  - 쿼리 : `SELECT BIN(12), OCT(12), HEX(255), HEX('A');`
  - 결과 : '1100', '14', 'FF', '41'

INSERT

- INSERT
  - 문자열(str)의 pos 위치부터 len 길이만큼을 새로운 문자열(newstr)로 바꾼 문자열을 반환
- 형식
  - INSERT(str, pos, len, newstr)
- 실행
  - 쿼리 : `SELECT INSERT('황금가면', 1, 2, '돼지');`
  - 결과 : '돼지가면'

LEFT / RIGHT

- LEFT / RIGHT
  - 왼쪽(LEFT)이나 오른쪽(RIGHT)에서 지정한 글자수만큼 문자열을 반환

- 형식
  - LEFT(str, len) / RIGHT(str, len)

- 실행
  - 쿼리 : `SELECT LEFT('황금가면', 2), RIGHT('황금가면', 2);`
  - 결과 : '황금', '가면'

LOWER / UPPER

- LOWER / UPPER
  - 전달된 문자열을 소문자(LOWER)나 대문자(UPPER)로 변환한 결과를 반환
- 형식
  - LOWER(str) / UPPER(str)
- 실행
  - 쿼리 : SELECT LOWER('SPIDERMAN'), UPPER('ultraman');
  - 결과 : 'spiderman', 'ULTERMAN'

LPAD / RPAD

- LPAD / RPAD
  - 지정한 길이(len)가 되도록 왼쪽(LPAD)이나 오른쪽(RPAD)에 문자열(padstr)을 채운 결과를 반환
- 형식
  - LPAD(str, len, padstr) / RPAD(str, len, padstr)
- 실행
  - 쿼리 : SELECT LPAD('hi', 4, '**'), RPAD('hi', 4, '??');
  - 결과 : '**hi', 'hi??'

LTRIM / RTRIM / TRIM

- LTRIM / RTRIM / TRIM
  - 왼쪽(LTRIM), 오른쪽(RTRIM), 양쪽(TRIM)의 공백을 제거한 문자열을 반환

- 형식
  - LTRIM(str) / RTRIM(str) / TRIM(str)

- 실행
  - 쿼리 : SELECT LTRIM(' 왼쪽'), RTRIM('오른쪽 '), TRIM(' 양쪽 ');
  - 결과 : '왼쪽', '오른쪽', '양쪽'

REPEAT

- REPEAT
  - 지정한 횟수(count)만큼 문자열(str)을 반복한 결과를 반환
- 형식
  - REPEAT(str, count)
- 실행
  - 쿼리 : `SELECT REPEAT('MySQL만세', 3);`
  - 결과 : 'MySQL만세MySQL만세MySQL만세'

REPLACE

- REPLACE
  - 특정 문자열(from_str)을 모두 찾아서 새로운 문자열(to_str)로 치환한 결과를 반환
- 형식
  - REPLACE(str, from_str, to_str)
- 실행
  - 쿼리 : SELECT REPLACE('간장 공장 공장장은 강 공장장이다', '공장', '농장');
  - 결과 : '간장 농장 농장장은 강 농장장이다'

REVERSE

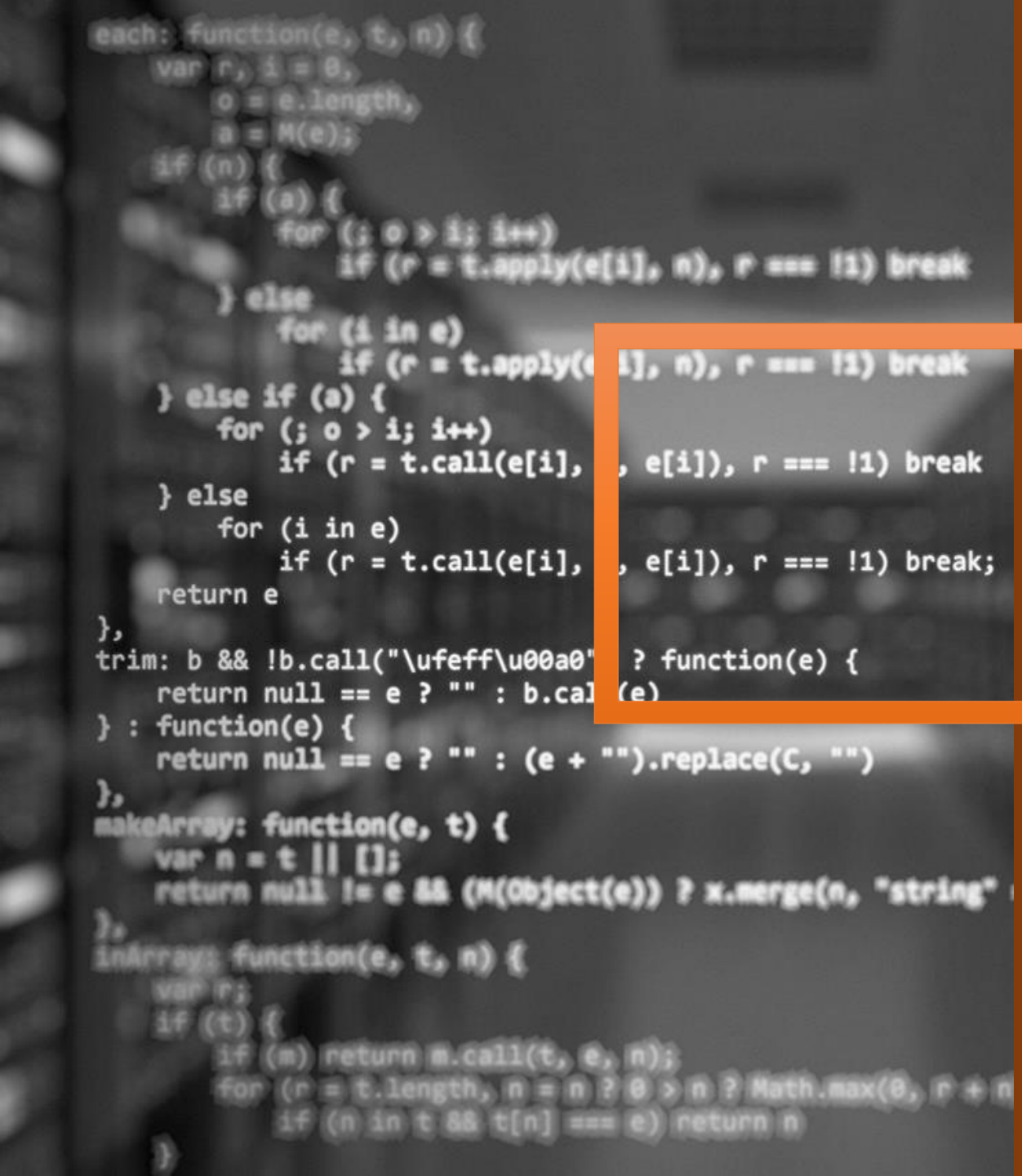
- REVERSE
  - 반전 결과를 반환
- 형식
  - REVERSE(str)
- 실행
  - 쿼리 : `SELECT REVERSE('abcd');`
  - 결과 : 'dcba'

SPACE

- SPACE
  - 지정한 만큼(N)의 공백 문자열을 반환
- 형식
  - SPACE(N)
- 실행
  - 쿼리 : `SELECT SPACE(6);`
  - 결과 : `'      '`

SUBSTRING

- SUBSTRING
  - 문자열의 pos 위치부터 len 길이만큼의 일부 문자열을 반환
- 형식
  - SUBSTRING(str, pos, len)
- 실행
  - 쿼리 : `SELECT SUBSTRING('hello world', 7), SUBSTRING('hello world', 7, 5);`
  - 결과 : 'world', 'world'
    - 위치는 1부터 시작한다.
    - 길이를 생략하면 문자열의 끝까지 반환한다.



2. 수학 함수

ABS

- ABS
  - 절대값을 반환
- 형식
  - ABS(X)
- 실행
  - 쿼리 : SELECT ABS(2), ABS(-2);
  - 결과 : 2, 2
    - X가 NULL이면 NULL을 반환한다.

CEILING / FLOOR

- CEILING / FLOOR
  - 정수 올림값(CEILING)과 정수 내림값(FLOOR)을 반환
- 형식
  - CEILING(X) / FLOOR(X)
- 실행
  - 쿼리 : SELECT CEILING(1.23), CEILING(-1.23), FLOOR(1.23), FLOOR(-1.23);
  - 결과 : 2, -1, 1, -2
    - X가 NULL이면 NULL을 반환한다.
    - CEILING() 함수는 CEIL() 함수와 같다.

MOD

- MOD
  - N을 M으로 나눈 나머지 값을 반환
- 형식
  - MOD(N, M) / N MOD M / N % M
- 실행
  - 쿼리 : SELECT MOD(234, 10), MOD(34.5, 3);
  - 결과 : 4, 1.5
    - N이나 M이 NULL이면 NULL을 반환한다.
    - M이 0이면 NULL을 반환한다.

PI

- PI
  - 원주율(π, pi) 값을 반환
- 형식
  - PI()
- 실행
  - 쿼리 : SELECT PI(), PI() + 0.0000000000000000;
  - 결과 : 3.141593, 3.14159265358793000

POW

- POW
  - X를 Y 제곱한 값을 반환
- 형식
  - POW(X, Y)
- 실행
  - 쿼리 : SELECT POW(2, 2), POW(2, -2);
  - 결과 : 4, 0.25
    - X나 Y가 NULL이면 NULL을 반환한다.
    - POW() 함수는 POWER() 함수와 같다.

RAND

- RAND
  - 0 <= v < 1.0 사이의 실수 난수를 반환
- 형식
  - RAND() / RAND(N)
- 실행
  - 쿼리 : SELECT RAND(), RAND(2), RAND(), RAND(2);
  - 결과 : 0.61788728518214 75, 0.6555866465490187, 0.8686402569232884, 0.6555866465490187
    - RAND(N) 함수는 N에 따라 매번 동일한 값을 반환한다.
  - 쿼리 : SELECT FLOOR(7 + RAND() * 5);
  - 결과 : 10
    - FLOOR(i + RAND() * (j - i)) 함수식은 i <= R < j 사이의 정수 난수를 반환한다.

ROUND

- ROUND
  - 반올림 값을 반환

- 형식
  - ROUND(X) / ROUND(X, D)

- 실행
  - 쿼리 : SELECT ROUND(-1.23), ROUND(-1.58), ROUND(1.58);
  - 결과 : -1, -2, 2
  - 쿼리 : SELECT ROUND(1.298, 1), ROUND(1.298, 0), ROUND(23.298, -1);
  - 결과 : 1.3, 1, 20
  - 쿼리 : SELECT ROUND(150.000, 2), ROUND(150, 2);
  - 결과 : 150.00, 150

SIGN

- SIGN
  - 인자의 부호에 따라 -1, 0, or 1을 반환
- 형식
  - SIGN(X)
- 실행
  - 쿼리 : SELECT SIGN(-2), SIGN(0), SIGN(2);
  - 결과 : -1, 0, 1
    - X가 NULL이면 NULL을 반환한다.

SQRT

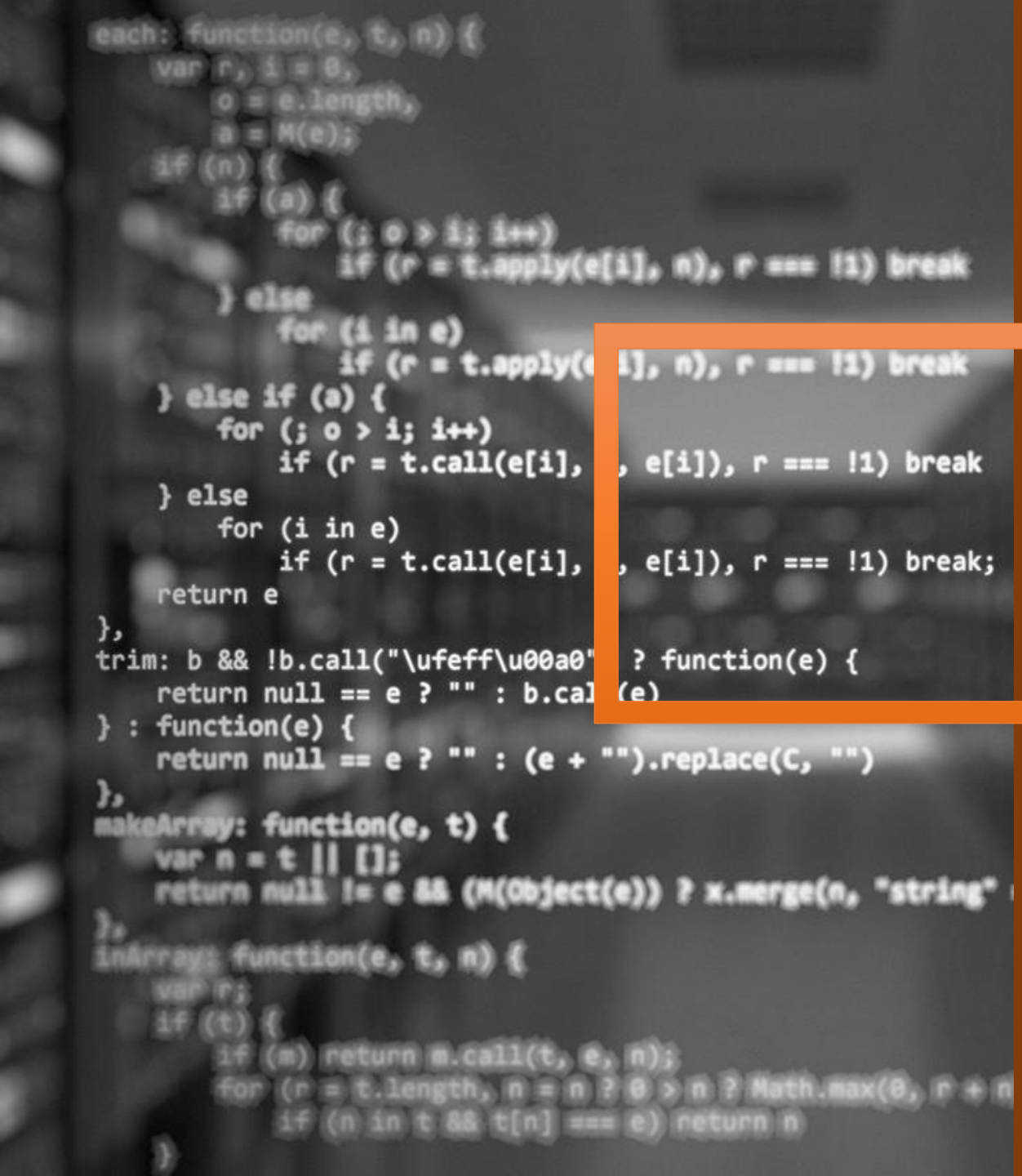
- SQRT
  - 제곱근(Square root) 값을 반환
- 형식
  - SQRT(X)
- 실행
  - 쿼리 : SELECT SQRT(4), SQRT(20), SQRT(-16);
  - 결과 : 2, 4.4721359549996, NULL
    - X가 음수 또는 NULL이면 NULL을 반환한다.

TRUNCATE

- TRUNCATE(X, D)
  - 절사 값을 반환

- 형식
  - TRUNCATE(X, D)

- 실행
  - 쿼리 : SELECT TRUNCATE(1.999, 1), TRUNCATE(1.999, 0), TRUNCATE(-1.999, 1);
  - 결과 : 1.9, 1, -1.9
  - 쿼리 : SELECT TRUNCATE(1999, -1), TRUNCATE(1999, -2), TRUNCATE(1999, -3);
  - 결과 : 1990, 1900, 1000



3. 날짜/시간 함수

CURDATE

- CURDATE
  - 현재 날짜를 'YYYY-MM-DD'(문자열) 또는 YYYYMMDD(숫자) 형식으로 반환

- 형식
  - CURDATE()

- 실행
  - 쿼리 : SELECT CURDATE(), CURDATE() + 0;
  - 결과 : '2008-06-13', 20080613
    - CURRENT_DATE, CURRENT_DATE() 함수와 동일하다.

CURTIME

- CURTIME
  - 현재 시간을 'hh:mm:ss'(문자열) 또는 hhmmss(숫자) 형식으로 반환

- 형식
  - CURTIME() / CURTIME(fsp)

- 실행
  - 쿼리 : SELECT CURTIME(), CURTIME(6), CURTIME() + 0;
  - 결과 : '12:34:21', '12:34:21.128496', 123421
    - CURRENT_TIME, CURRENT_TIME() 함수와 동일하다.
    - fsp : 초 단위 정밀도, 0 ~ 6 사이 값 사용이 가능하다.

NOW

- NOW
  - 현재 날짜시간을 'YYYY-MM-DD hh:mm:ss'(문자열) 또는 YYYYMMDDhhmmss(숫자) 형식으로 반환

- 형식
  - NOW() / NOW(fsp)

- 실행
  - 쿼리 : SELECT NOW(), NOW(6), NOW() + 0;
  - 결과 : '2008-06-13 12:34:21', '2008-06-13 12:34:21.128496', 20080613123421
    - CURRENT_TIMESTAMP, CURRENT_TIMESTAMP() 함수와 동일하다.
    - fsp : 초 단위 정밀도, 0 ~ 6 사이 값 사용이 가능하다.

SYSDATE

- SYSDATE
  - 현재 날짜시간을 'YYYY-MM-DD hh:mm:ss'(문자열) 또는 YYYYMMDDhhmmss(숫자) 형식으로 반환

- 형식
  - SYSDATE() / SYSDATE(fsp)

- 실행
  - 쿼리 : SELECT SYSDATE(), SLEEP(2), SYSDATE();
  - 결과 : '2008-06-13 12:34:21', '2008-06-13 12:34:23'
    - SYSDATE() 함수는 함수가 실행된 시간을 사용한다. (2초 차이 발생)
  - 쿼리 : SELECT NOW(), SLEEP(2), NOW();
  - 결과 : '2008-06-13 12:34:21', '2008-06-13 12:34:21'
    - NOW() 함수는 쿼리문이 실행된 시간을 사용한다. (시간 차이 없음)

DATE / TIME

- DATE / TIME
  - 날짜시간에서 날짜(DATE), 시간(TIME) 부분만 추출해서 반환
- 형식
  - DATE(expr) / TIME(expr)
- 실행
  - 쿼리 : `SELECT DATE('2003-12-31 01:02:03.000123'), TIME('2003-12-31 01:02:03.000123');`
  - 결과 : '2003-12-31', '01:02:03.000123'

DATEDIFF / TIMEDIFF

- DATEDIFF / TIMEDIFF
  - expr1 - expr2 값을 day 단위(DATEDIFF) 또는 시간형식(TIMEDIFF) 으로 반환
- 형식
  - DATEDIFF(expr1, expr2) / TIMEDIFF(expr1, expr2)
- 실행
  - 쿼리 : `SELECT DATEDIFF('2007-12-31 23:59:59', '2007-12-30');`
  - 결과 : 1
  - 쿼리 : `SELECT TIMEDIFF('2008-12-31 23:59:59.000001', '2008-12-30 01:01:01.000002');`
  - 결과 : '46:58:57.999999'

ADDDATE / DATE_ADD

- ADDDATE / DATE_ADD
  - 특정 기간 이후 날짜시간을 반환

- 형식
  - ADDDATE(date, INTERVAL expr unit) == DATE_ADD(date, INTERVAL expr unit)
  - ADDDATE(date, days)

- 실행
  - 쿼리 : SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY), ADDDATE('2008-01-02', 31),
    ADDDATE('2008-01-02 23:59:59', INTERVAL 1 SECOND);
  - 결과 : '2008-02-02', '2008-02-02', '2008-01-03 00:00:00'
  - 쿼리 : SELECT DATE_ADD('2008-01-02', INTERVAL 31 DAY);
  - 결과 : '2008-02-02'
    - unit : YEAR, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND, MICROSECOND 등

SUBDATE / DATE_SUB

- SUBDATE / DATE_SUB
  - 특정 기간 이전 날짜/시간을 반환

- 형식
  - SUBDATE(date, INTERVAL expr unit) == DATE_SUB(date, INTERVAL expr unit)
  - SUBDATE(date, days)

- 실행
  - 쿼리 : SELECT SUBDATE('2008-01-02', INTERVAL 31 DAY), SUBDATE('2008-01-02', 31),
    SUBDATE('2008-01-02 00:00:00', INTERVAL 1 SECOND);
  - 결과 : '2007-12-02', '2007-12-02', '2008-01-01 23:59:59'
  - 쿼리 : SELECT DATE_SUB('2008-01-02', INTERVAL 31 DAY);
  - 결과 : '2007-12-02'
    - unit : YEAR, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND, MICROSECOND 등

DATE_FORMAT

- DATE_FORMAT
  - 날짜시간을 특정 형식(format)의 문자열로 반환
- 형식
  - DATE_FORMAT(date, format)
- 실행
  - 쿼리 : SELECT DATE_FORMAT(NOW(), '%Y-%m-%d %a %p %h:%i:%s');
  - 결과 : '2009-10-04 Sun PM 09:35:28'

종류	의미	결과
%y	년도	09
%Y	년도	2009
%M	월 이름	January..December
%b	월 약어	Jan..Dec
%c	월	0..12
%m	월	00..12
%W	일 이름	Sunday..Saturday
%a	일 약어	Sun..Sat
%e	일	0..31
%d	일	00..31
%p	AM or PM	AM or PM
%H	시 (24-hour)	00..23
%h %I	시 (12-hour)	01..12
%k	시 (24-hour)	0..23
%l	시 (12-hour)	1..12
%i	분	00..59
%s %S	초	00..59
%f	마이크로초	000000..999999
%T	24-hour TIME	hh:mm:ss
%r	12-hour TIME	AM/PM hh:mm:ss

STR_TO_DATE

- STR_TO_DATE
  - 문자열을 지정한 형식(format)으로 해석한 날짜시간을 반환
- 형식
  - STR_TO_DATE(str, format)
- 실행
  - 쿼리 : `SELECT STR_TO_DATE('01,5,2013', '%d,%m,%Y');`
  - 결과 : '2013-05-01'

종류	의미	결과
%y	년도	09
%Y	년도	2009
%M	월 이름	January..December
%b	월 약어	Jan..Dec
%c	월	0..12
%m	월	00..12
%W	일 이름	Sunday..Saturday
%a	일 약어	Sun..Sat
%e	일	0..31
%d	일	00..31
%p	AM or PM	AM or PM
%H	시 (24-hour)	00..23
%h %I	시 (12-hour)	01..12
%k	시 (24-hour)	0..23
%l	시 (12-hour)	1..12
%i	분	00..59
%s %S	초	00..59
%f	마이크로초	000000..999999
%T	24-hour TIME	hh:mm:ss
%r	12-hour TIME	AM/PM hh:mm:ss

YEAR / MONTH / DAY

YEAR / MONTH / DAY

- 날짜에서 년(YEAR), 월(MONTH), 일(DAY)을 반환

형식

- YEAR(date) / MONTH(date) / DAY(date)

실행

- 쿼리 : `SELECT YEAR('2007-02-03'), MONTH('2007-02-03'), DAY('2007-02-03');`
- 결과 : 2007, 2, 3
  - DAY() 함수는 DAYOFMONTH() 함수와 같다.

HOUR / MINUTE / SECOND / MICROSECOND

HOUR / MINUTE / SECOND / MICROSECOND

- 시간에서 시(HOUR), 분(MINUTE), 초(SECOND), 마이크로초(MICROSECOND)를 반환

형식

- HOUR(time) / MINUTE(time) / SECOND(time) / MICROSECOND(expr)

실행

- 쿼리 :
```
SELECT   HOUR('2007-02-03 10:05:03'),
         MINUTE('2007-02-03 10:05:03'),
         SECOND('2007-02-03 10:05:03'),
         MICROSECOND('12:00:00.123456');
```
- 결과 : 10, 5, 3, 123456

EXTRACT

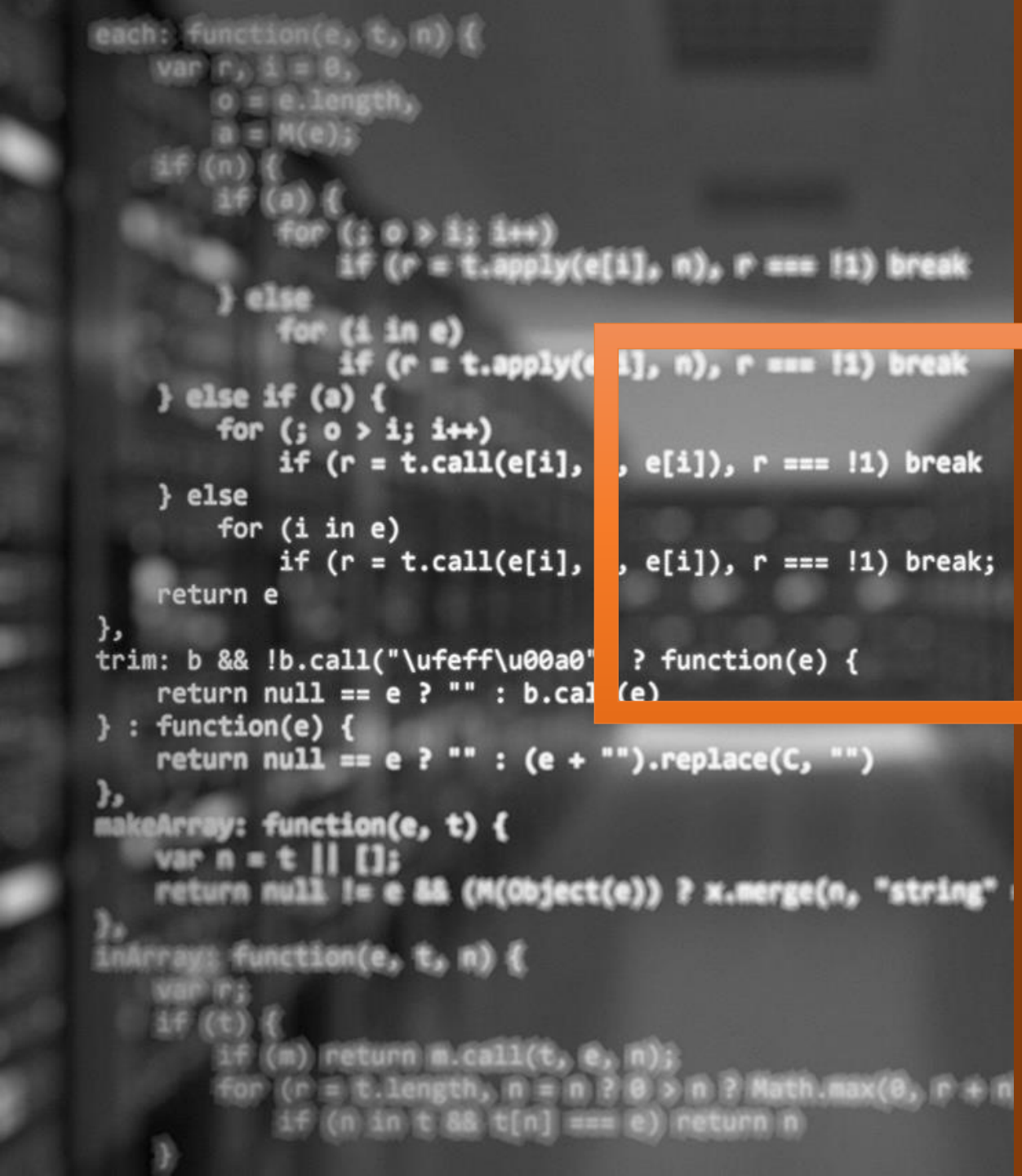
- EXTRACT
  - 날짜시간에서 지정한 unit을 추출해서 반환

- 형식
  - EXTRACT(unit FROM date)

- 실행
  - 쿼리 : SELECT EXTRACT(YEAR FROM '2007-02-03');
  - 결과 : 2007
    - unit : YEAR, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND, MICROSECOND 등

MAKEDATE / MAKETIME

- MAKEDATE / MAKETIME
  - 날짜(MAKEDATE)나 시간(MAKETIME)을 만든 뒤 반환
- 형식
  - MAKEDATE(year, dayofyear) / MAKETIME(hour, minute, second)
- 실행
  - 쿼리 : SELECT MAKEDATE(2011, 31), MAKEDATE(2011, 32), MAKEDATE(2011, 365);
  - 결과 : '2011-01-31', '2011-02-01', '2011-12-31'
  - 쿼리 : SELECT MAKETIME(12, 15, 30);
  - 결과 : '12:15:30'


each: function(e, t, n) {
var r, i = 0,
o = e.length,
a = M(e);
if (n) {
if (a) {
for (; o > i; i++)
if (r = t.apply(e[i], n), r === !1) break
} else
for (i in e)
if (r = t.apply(e[i], n), r === !1) break
} else if (a) {
for (; o > i; i++)
if (r = t.call(e[i], e[i]), r === !1) break
} else
for (i in e)
if (r = t.call(e[i], e[i]), r === !1) break;
return e
},
trim: b && !b.call("\ufeff\u00a0") ? function(e) {
return null == e ? "" : b.call(e)
} : function(e) {
return null == e ? "" : (e + "").replace(C, "")
},
makeArray: function(e, t) {
var n = t || [];
return null != e && (M(Object(e)) ? x.merge(n, "string"
},
inArray: function(e, t, n) {
var r;
if (t) {
if (m) return m.call(t, e, n);
for (r = t.length, n = n ? 0 > n ? Math.max(0, r + n
if (n in t && t[n] === e) return n
}

4. 기타 함수

IF

- IF
  - expr1 이 true 이면 expr2 반환, 아니면 expr3 반환
- 형식
  - IF(expr1, expr2, expr3)
- 실행
  - 쿼리 : `SELECT IF(1 > 2, 2, 3), IF(1 < 2, 'yes', 'no');`
  - 결과 : 3, 'yes'

CASE

CASE Operator

- case_value 에 따라 실행할 구문을 선택하는 연산자 (함수는 아님)

형식

- CASE case_value
  WHEN when_value THEN statement_list
  [WHEN when_value THEN statement_list] ...
  [ELSE statement_list]
  END
- CASE
  WHEN search_condition THEN statement_list
  [WHEN search_condition THEN statement_list] ...
  [ELSE statement_list]
  END

실행

- 쿼리 :
```
SELECT
    CASE 1
      WHEN 1 THEN 'one'
      WHEN 2 THEN 'two'
      ELSE 'more'
    END;
```
- 결과 : 'one'
- 쿼리 :
```
SELECT
    CASE
      WHEN 1 > 0 THEN 'TRUE'
      ELSE 'FALSE'
    END;
```
- 결과 : 'TRUE'

IFNULL

- IFNULL
  - expr1 이 NULL 이면 expr2 반환, 아니면 expr1 반환
- 형식
  - IFNULL(expr1, expr2)
- 실행
  - 쿼리 : SELECT IFNULL(1, 0), IFNULL(NULL, 10);
  - 결과 : 1, 10