제09장

인덱스와 뷰

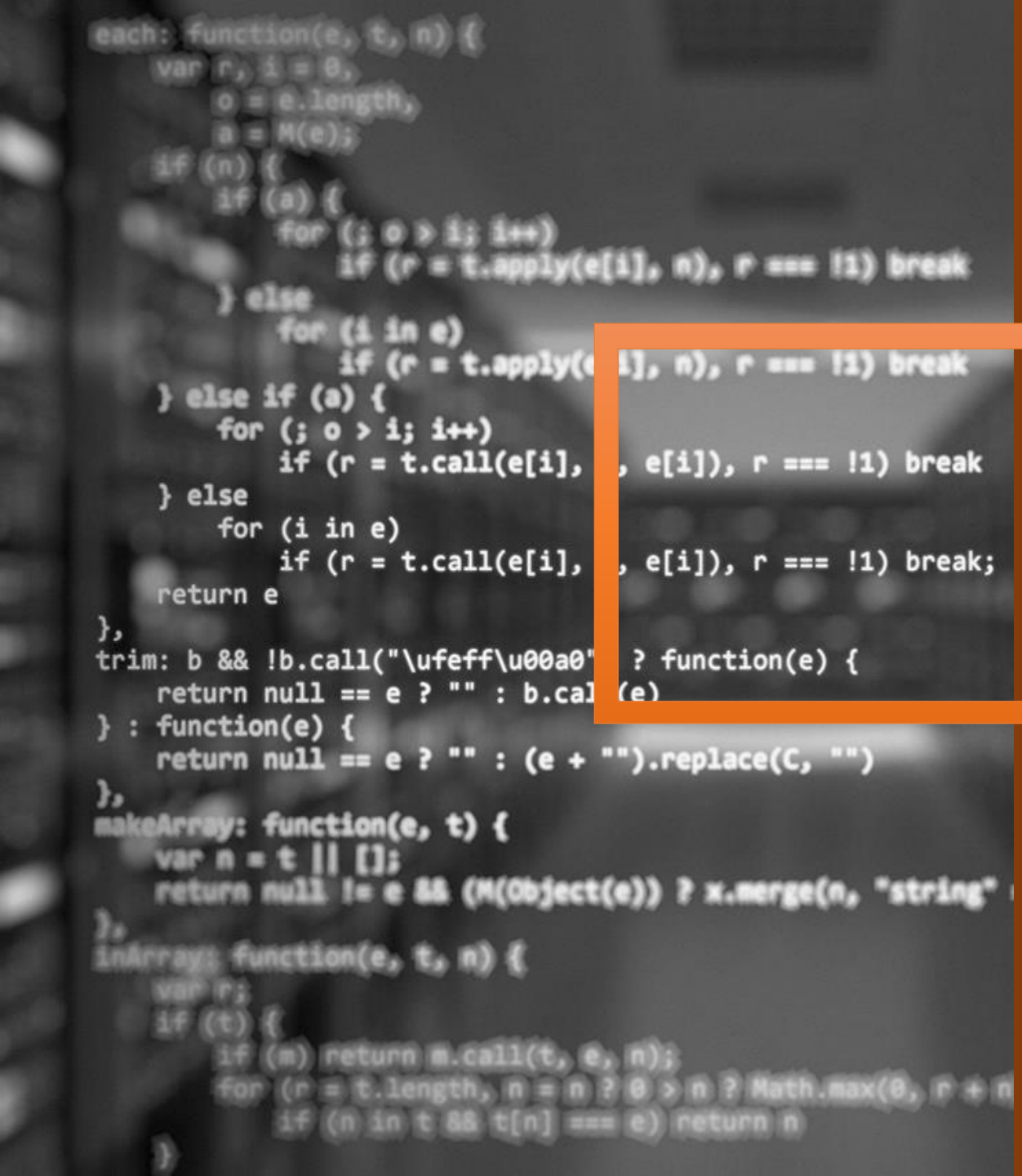
MySQL

학습목표

1. 인덱스에 대해서 알 수 있다.
2. 뷰에 대해서 알 수 있다.

목차

1. 인덱스
2. 뷰


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
}

1. 인덱스

인덱스

인덱스

- Index
- SQL 문의 처리 속도 향상을 위해 특정 칼럼을 대상으로 생성하는 데이터베이스 객체

인덱스 장점

- 검색 속도가 굉장히 빨라질 수 있음
- 검색 쿼리의 부하가 줄어들면 곧 시스템 전체 성능이 향상됨

인덱스 단점

- 인덱스를 별도로 저장해야 하기 때문에 DB 크기의 10% 정도 추가 공간이 필요함
- 변경 작업(INSERT, UPDATE, DELETE)이 자주 발생하면 오히려 성능이 떨어질 수 있음

클러스터 인덱스

- **클러스터 인덱스 (Primary Index)**
  - Clustered Index
  - 테이블당 1개만 존재할 수 있는 인덱스
  - 데이터들을 특정 기준으로 자동으로 정렬해 주는 인덱스 (정렬되어 있는 영어사전)
  - 인덱스 저장을 위한 공간을 적게 사용함
  - 보조 인덱스에 비해 검색 속도가 더 빠르나 변경(삽입/수정/삭제) 속도가 더 느림
  - 기본키(Primary Key)로 설정된 칼럼에는 자동으로 클러스터 인덱스가 생성됨
    - ➢ 기본키로 설정된 칼럼으로 조회 조건을 만드는 것이 검색 속도를 높이는 방법임

보조 인덱스

보조 인덱스 (Secondary Index)

- Non-Clustered Index
- 한 테이블에 여러 개가 존재할 수 있는 인덱스
  - 그렇다고 여러 개를 만들면 오히려 성능이 더 떨어질 수 있음
- 클러스터 인덱스와 달리 데이터를 자동으로 정렬하지 않음
- 클러스터 인덱스에 비해 검색 속도는 더 느리지만 변경(삽입/수정/삭제) 속도가 덜 느림
- UNIQUE 키워드로 고유키를 지정하면 자동으로 보조 인덱스가 생성됨
- CREATE INDEX 문으로 직접 보조 인덱스 생성 가능

인덱스 쿼리문

인덱스 생성 구문

```
CREATE [UNIQUE] INDEX 인덱스_이름
    ON 테이블_이름(칼럼_이름) [ASC | DESC]
```

인덱스 삭제

- 기본키나 고유키로 자동 생성된 인덱스는 삭제할 수 없음
- 기본키나 고유키 제약조건을 삭제하면 자동으로 생성된 인덱스도 삭제됨

```
DROP INDEX 인덱스_이름
    ON 테이블_이름
```

인덱스 쿼리문

인덱스 적용 구문

- CREATE INDEX 문으로 보조 인덱스를 생성한 경우에는 인덱스 적용 구문이 추가로 필요함

```
ANALYZE TABLE 테이블_이름
```

인덱스 확인

```
SHOW INDEX FROM 테이블_이름
```

인덱스 활용

테이블 상황

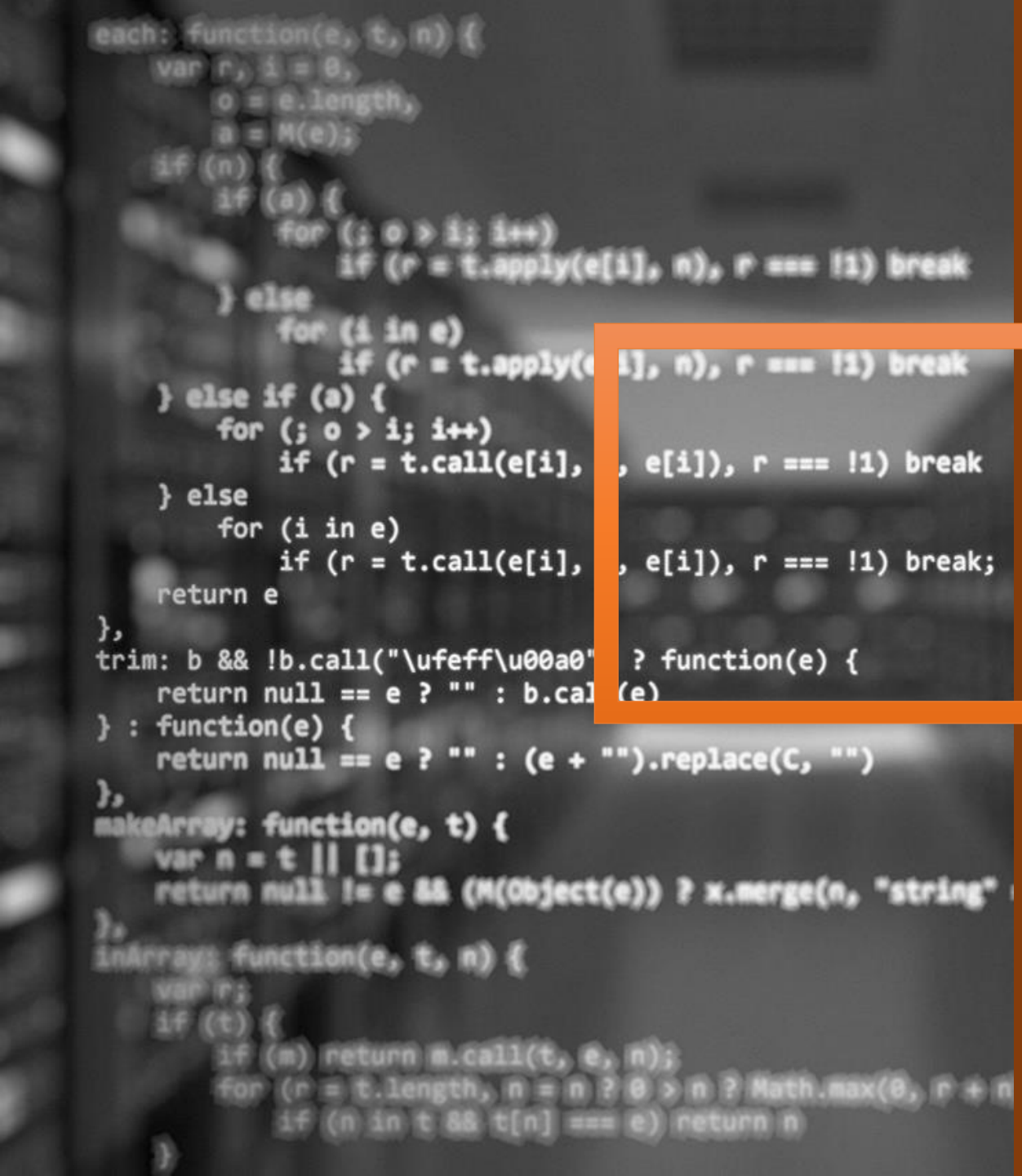
- mem_id 칼럼은 PK로 지정되어 클러스터 인덱스를 가짐
- mem_id 칼럼은 1부터 1씩 증가하는 값을 가짐
- 테이블에는 1000개 정도의 행(Row)이 존재함

Good 😌

- WHERE mem_id = 1;
  - ➢ Single Row
- WHERE mem_id >= 500;
  - ➢ Index Range Scan

Bad 😥

- WHERE mem_name = 'kim';
  - ➢ Full Table Scan (인덱스 미사용)
- WHERE mem_id >= 1;
  - ➢ Full Table Scan (MySQL이 판단)
- WHERE mem_id * 2 >= 1000;
  - ➢ Full Table Scan (인덱스 칼럼에 연산)


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
}

2. 뷰

뷰

뷰

- View
- 하나 이상의 기본 테이블이나 다른 뷰(View)를 이용하여 생성하는 가상 테이블
- 실제 테이블처럼 행과 열로 구성되지만 실제로 데이터를 가지지는 않음
- 디스크에 저장한 테이블이 아닌 쿼리만 저장해 둔 형태를 가짐
- 생성된 뷰는 SELECT 문을 활용해서 일반 테이블처럼 조회할 수 있음

뷰의 장단점

- 장점
  - 특정 사용자에게 보고자 하는 데이터만 제공하는 편의성 제공
  - 복잡한 쿼리를 단순하게 호출
  - 쿼리문의 재사용 가능

- 단점
  - 정의된 뷰의 변경이 불가능
  - 삽입, 삭제, 갱신 작업에 많은 제약이 따름
  - 자신의 인덱스를 가질 수 없음

뷰 쿼리문

뷰 생성 구문

- OR REPLACE 구문을 추가하면 기존 뷰를 대체할 수 있음
- 기존 뷰가 없는 경우 CREATE VIEW 와 동일하게 동작함

```
CREATE [OR REPLACE] VIEW 뷰_이름 AS
   SELECT 조회_칼럼
   FROM 테이블_이름
   WHERE  조건식
```

뷰 삭제

```
DROP VIEW 뷰_이름
```

뷰 활용

- tbl_product 테이블

code	model	category	price	amount	manufactured
A1	WING	MAIN	200	2	20/01/01
A2	LOCK	SUB	300	1	20/02/01
B1	WHEEL	SUB	100	4	20/03/01
B2	ARM	MAIN	50	2	20/04/01
C1	BODY	MAIN	500	1	20/05/01
C2	HEAD	SUB	400	1	20/06/01

- category='MAIN' 인 제품을 조회하는 뷰 생성

```
CREATE OR REPLACE VIEW v_main AS
    SELECT code, model, category, price, amount, manufactured FROM tbl_product
    WHERE category = 'MAIN';
```

- 뷰 조회

```
SELECT code, model, category, price, amount, manufactured FROM v_main;
```

code	model	category	price	amount	manufactured
A1	WING	MAIN	200	2	20/01/01
B2	ARM	MAIN	50	2	20/04/01
C1	BODY	MAIN	500	1	20/05/01