이문희

DATABASE SYSTEM IMPLEMENTATION

이문희

DATABASE SYSTEM IMPLEMENTATION

소개

- Book

  H. Garcia-Molina, J. Ullman, J. Widom, *Database System Implementation*, Prentice Hall, 2000.

내용

1. Introduction & Data Storage (1주)
2. Representing Data Elements (2, 4주)
3. Index Structures (5, 6주)
4. Query Execution (7,8주)
5. The Query Compiler (9, 10주)
6. Coping with System Failures (11, 12주)
7. Concurrency Control (13, 14, 15주)
8. Multidimensional Indexes (16, 17주)
9. Distributed DBMS (18주)

1. INTRODUCTION & DATA STORAGE

Table of Contents

Capabilities of DBMS

- 영속적인 Storage 제공
  - File system의 기능
  - Flexibility (data independence) 제공
- Programming Interface
  - File system보다 user convenient
- Transaction Management
  - ACID property 제공

Overview of Query Processing

- 사용자 요청이 DB에 접근하는 두 가지 경로
  - Answering the query
    1. Query compiler에 의해 파싱, 최적화 (결과 값은 query plan 으로 나타남)
    2. *Execution engine*으로 전달
    3. 데이터 파일(포맷과 사이즈)과 인덱스 파일 정보를 알고 있는 *resource manager*에게 레코드를 요청한다
    4. page로 translate되고 *buffer manager*에게 요청이 전달됨
    5. *buffer manager*는 데이터를 얻기 위해서 storage manager과 통신한다

  - Transaction processing (query의 모음)
    - *Concurrency-control manager*(*scheduler*)
      - Atomicity, Isolation
    - *Logging and recovery manager*
      - Durability

Main-memory Buffers and the Buffer Manager

- DBMS의 모든 컴포넌트는 정보를 요구
  - 직접적으로 얻거나 버퍼 관리자를 통해 획득
  - 정보의 종류
    - 데이터: DB자체의 내용물
    - 메타데이터: DB 스키마(DB의 구조, 제약조건)
    - 통계치: 데이터에 대한 속성(DB컴포넌트 혹은 relation의 크기 , 값 등)
    - 인덱스

Transaction Processing

- *A unit of work*

- Transaction manager ⇔ User(Application)
  - Transaction commands
    - 예상 결과물에 대한 정보, 트랜잭션의 시작과 끝

- 트랜잭션 매니저가 수행하는 일
  1. Logging: durability를 보장
     1. log manager:
        로그에 변화를 기록한 다음, 버퍼매니저에게 버퍼가 디스크에 쓰여졌는지 확인
     2. recovery manager:
        로그를 시험 & consistent 상태로 복원
  2. Concurrency control
     여러 트랜잭션이 동시에 실행되어도 각각 실행 된 것처럼 … (lock을 이용)
  3. Deadlock resolution
     동치관계를 감지하여 한쪽을 abort

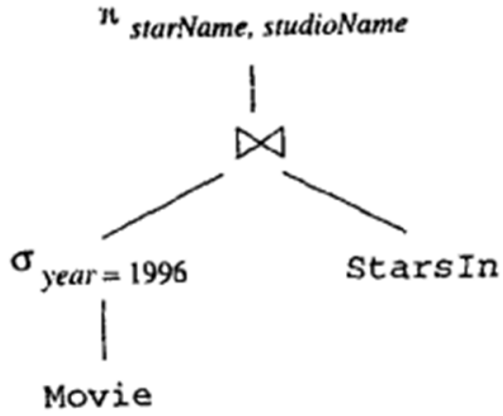
Query Processor

- 쿼리 컴파일러
  - 쿼리 플랜 → 관계대수 (relation algebra) 나타냄
  - 쿼리 파서
    - Parse tree 생성
  - 쿼리 전처리기
    - 시맨틱 체크(<예> 실제로 테이블이 존재하나?)
    - 대수 트리로 표현
  - 쿼리 옵티마이저
    - 대수 트리를 최적화
- Execution Engine

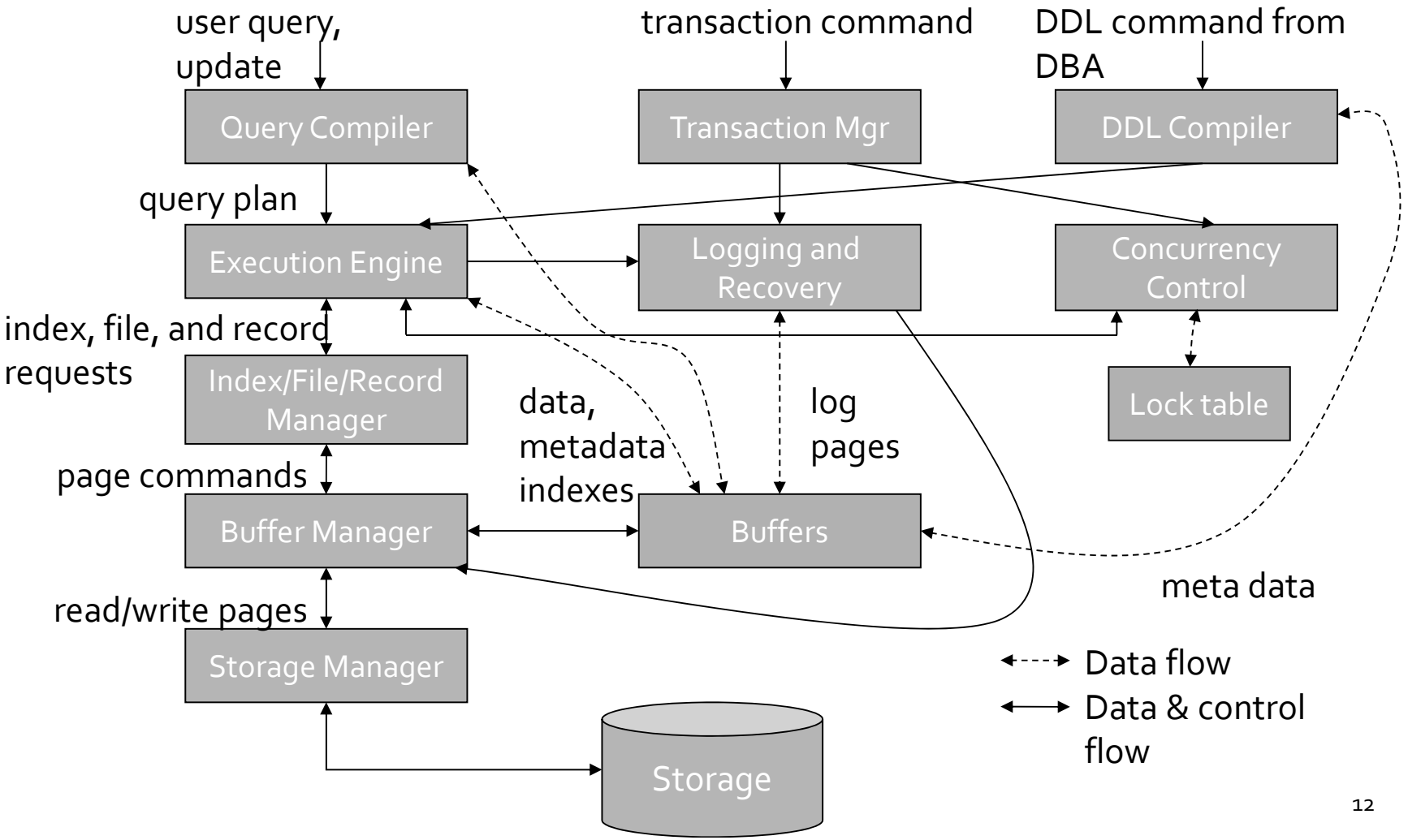
```
StarsIn(title, year, starName)
Movie(title, year, length, inColor, studioName, producerC#)
```

```
CREATE VIEW MoviesOf1996 AS
    SELECT *
    FROM Movie
    WHERE year = 1996;
```

```
SELECT starName, studioName
FROM MoviesOf1996 NATURAL JOIN StarsIn;
```



1. DBMS Components



DATA STORAGE

Table of Contents

1. Disks

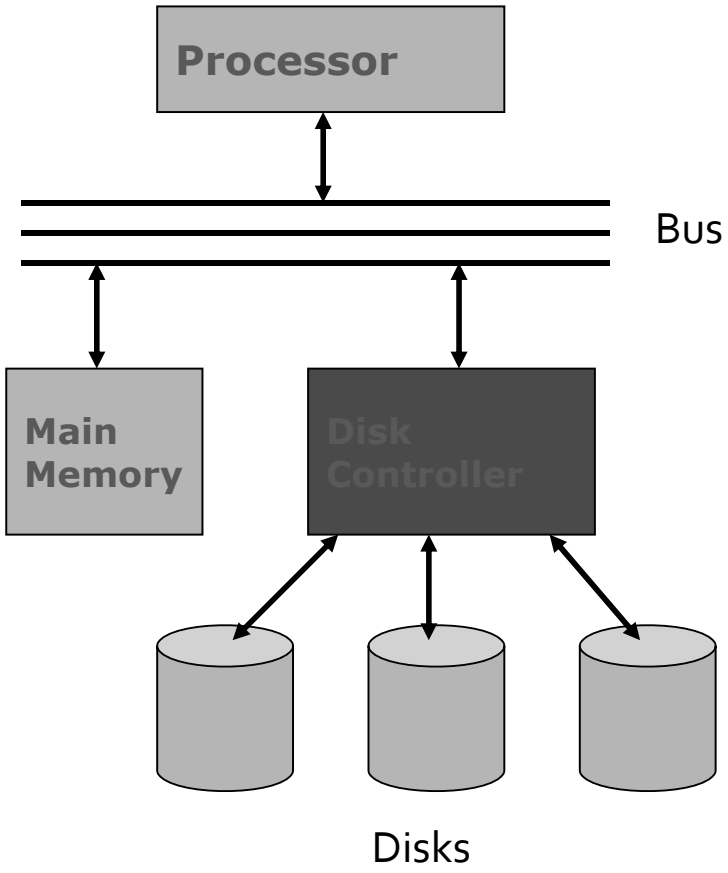
- Sector VS. Block
- The Disk Controller
- SSD

Sectors vs. Blocks

- Sector
  - Physical unit of the disk
  - Unit of error correction/detection
  - Parity-check bit is chosen so number of 1's is even.
    - Single errors detectable
  - A bad sector is "cancelled" by the disk controller.

- Block
  - Logical unit of data that are transferred between disk and main memory
  - Typical: one 4K block = 8 512-byte sectors.

The Disk Controller



- Buffer data in and out of disk.
- Schedule the disk heads.
- Manage the "bad blocks" so they are not used.

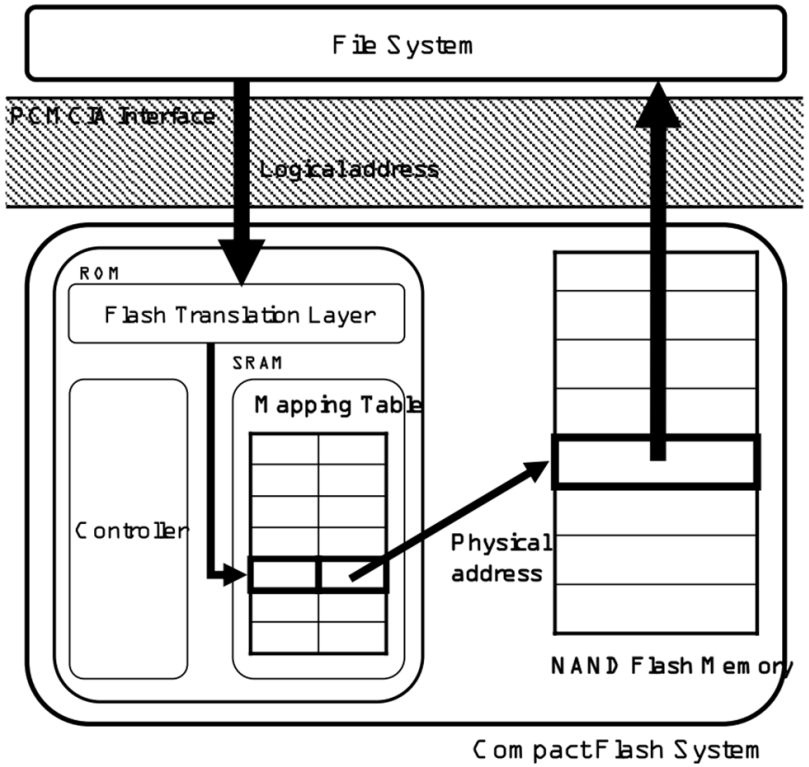
SSD - flash memory(NAND)

Category	Mini Hard-Disk	NAND Flash
Read/Write Unit	Sector (512bytes)	Sector (512bytes)
Mechanical Moving Parts	Spinning disk head (seek + latency)	No (all electronics)
Energy Consumption	High	Low
Shock Resistance	Bad	Good
Density	Has reached limit	Room for improvement
Endurance	1,000,000 overwrites	10-100,000 erases/cell
Random Read/Write	High (in mili-seconds)	Low (in tens of micro-seconds)
Sequential Read/Write	High bandwidth	High bandwidth
Size Reduction	Has reached limit (0.85 inch)	Room for improvement
Noise & Vibration	High	No

Table 1. Comparison of the mini hard-disk and the NAND flash

2007년

SSD - flash memory(NAND)



- FTL(Flash Translation Layer)
  - 제한된 쓰기 회수
    - Ware-leveling
  - 쓰기 ,지우기 속도 비대칭적
    - In-place update = 지우기 + 쓰기 (속도↓, 비용 ↑)
  - **Update = 다른 곳에 쓴 다음 매핑**
    - **Garbage Collection!**

2. Using Secondary Storage Efficiently

- The I/O Model of Computation
- Good DBMS Algorithms
- Two-Phase, Multiway Merge-Sort

I/O Model of Computation

- Disk-based➔Disk I/O
  - 블록의 기록/판독 연산은 메모리 상에서의 연산에 비해 매우 오버헤드가 큰 연산
  - 블록 액세스 수를 비교함으로써 알고리즘들간의 성능 평가 가능.

- 그렇다면 Flash는?
  - Update operator의 회수

Good DBMS Algorithm (SSD)

- Update가 적게 되는 알고리즘이면 좋다.

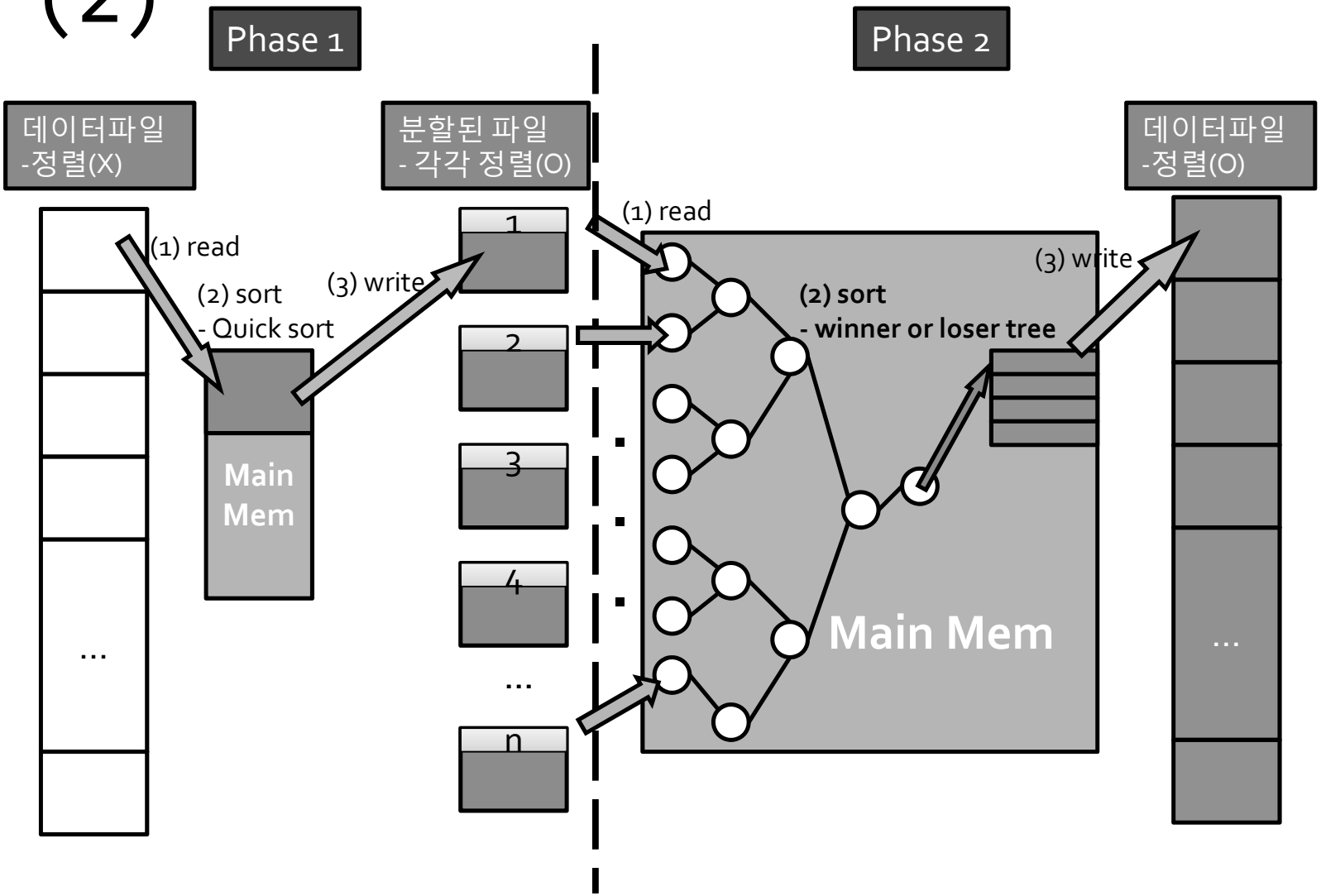
- 자주 사용되는 블록들은 메모리의 버퍼에 상주시키도록 노력하라.

Two-Phase, Multiway Merge Sort (1)

- 대용량의 데이터
  - Internal Sort 불가능
- binary sort/merge,
  balanced sort/merge
  balanced k-way sort/merge
  poly phase sort/merge

- **2PMMS: 2 * (reads + writes) per block**

Two-Phase, Multiway Merge Sort (2)



Prefetching/Double Buffering

- 기본 개념
  - 앞으로 액세스될 블록들을 미리 판독
  - 기록될 블록을 일정 시간 대기한 후, 여유 시간에 기록

- Application of Phase 2 of 2PMMS
  - 입력/출력 리스트마다 2개의 버퍼 할당
  - 입력 리스트에서 블록 판독 및 비교를 병행하여 처리
  - 출력 리스트에 대한 생성 및 기록을 병행하여 처리

3. Disk Failures

- Failure의 종류
- Checksums

Failure의 종류

- Intermittent Failure
  - Sector의 판독/기록시 발생하는 일시적인 연산 오류
- Media Decay
  - 하나 이상의 bit가 영구히 손실될 경우
- Write Failure
  - Sector를 기록하는 도중 오류 발생. Sector의 일부만 기록됨으로써 이전/이후 내용이 모두 손실.
- Disk Crash
  - 전체 디스크의 내용이 판독 불가

Checksums

- Parity
  - Even parity –
    전체 1의 개수가 짝수(1110:1, 1010:0)
  - Odd parity –
    전체 1의 개수가 홀수(1110:0,1010:1)
  - Parity bit의 수에 따라 error detection의 정확성 증가

4. Recovery from Disk Crashes

- ~~Mirroring as a Redundancy Technique (RAID 1)~~
- Parity Blocks (RAID 4)
- An Improvement: RAID 5
- RAID 6

Parity Blocks (RAID level 4)

- Use *n* data disks and one redundant disk.
  - Redundant disk의 각 block은 data disk의 나머지 block들에 대한 parity block의 역할.

- Example
  - *n* = 3, blocks are 8 bits.
  - First blocks of data disks: 11110000, 10101010, 00111000
  - The first block of the redundant disk is
    - 11110000 ⊕ 10101010 ⊕ 00111000 = 01100010
  - ⊕: Bitwise modulo-2 sum operation

Parity Blocks (RAID level 4)

- Reading
  - Read data disk normally.
  - 현재 사용 중인 Disk *i*에서 데이터를 판독할 경우, 다른 모든 디스크에서 데이터 판독 후 ⊕ 연산 실행.

- To write block *j* of data disk *i* (new value = *x*):
  - Read old value of that block, say *y*.
  - Read the *j*th block of the redundant disk, say *z*.
  - Compute $w = x \oplus y \oplus z$
  - Write *x* in block *j* of disk *i*.
  - Write *w* in block *j* of the redundant disk.
    - Total of four disk I/O's required.

Parity Blocks (RAID level 4)

- Failure Recovery
  - 임의의 하나의 디스크 고장 시 복구 가능
  - 복구 방법: 다른 디스크들의 블록들에 대해 ⊕ 계산

- Example
  - Disk 2 crashes. First blocks are:
    - 11110000, ????????, 00111000, 01100010.
  - Fill in ?'s by ⊕ applied to other three:
    - ???????? = 10101010

RAID Level 5

- RAID Level 4의 문제점
  - 블록 기록시, redundant disk가 bottleneck.

- 해결 방법
  - 블록들마다 redundant disk의 위치를 변경
  - Example: *n* disks
    - Block *j* is redundant on disk *i*, if *i* = remainder of *j* / *n*.

RAID Level 6

- Reading
  - Data disk에서 판독. Redundant disk는 무시.

- Writing
  - 해당 row의 position이 1인 디스크들과 ⊕ 계산
  - Example: 8-bit blocks,
    - 4 data disks with first blocks:
      - x_1 = 11110000, x_2 = 10101010
      - x_3 = 00111000, x_4 = 01000001
    - Redundant disks:
      - x_5 = 01100010 ($x_1 \oplus x_2 \oplus x_3$)
      - x_6 = 00011011 ($x_1 \oplus x_2 \oplus x_4$)
      - x_7 = 10001001 ($x_1 \oplus x_3 \oplus x_4$)

new x_2

00001111

RAID Level 6

- Handling Two Disk Failures
  - Disk *a*와 *b*의 고장을 가정.
  - Hamming matrix에서 column *a*와 *b*가 다른 row *r*을 발견.
    - Row *r*에서 column *a*는 1로 가정.
  - Row *r*을 이용하여 disk *a*의 내용을 복구
    - Column *b*의 값은 0이므로, 나머지 column들의 $\oplus$ 연산을 이용하여 *a*의 값 복구 가능.
  - Column *b*가 1인 row를 이용하여 disk *b*의 내용 복구.

- Example
  - Data Disk: 11110000, ???????, 00111000, 01000001
  - Redundant Disk: ????????, 10111110, 10001001

Thank you.

Q&A