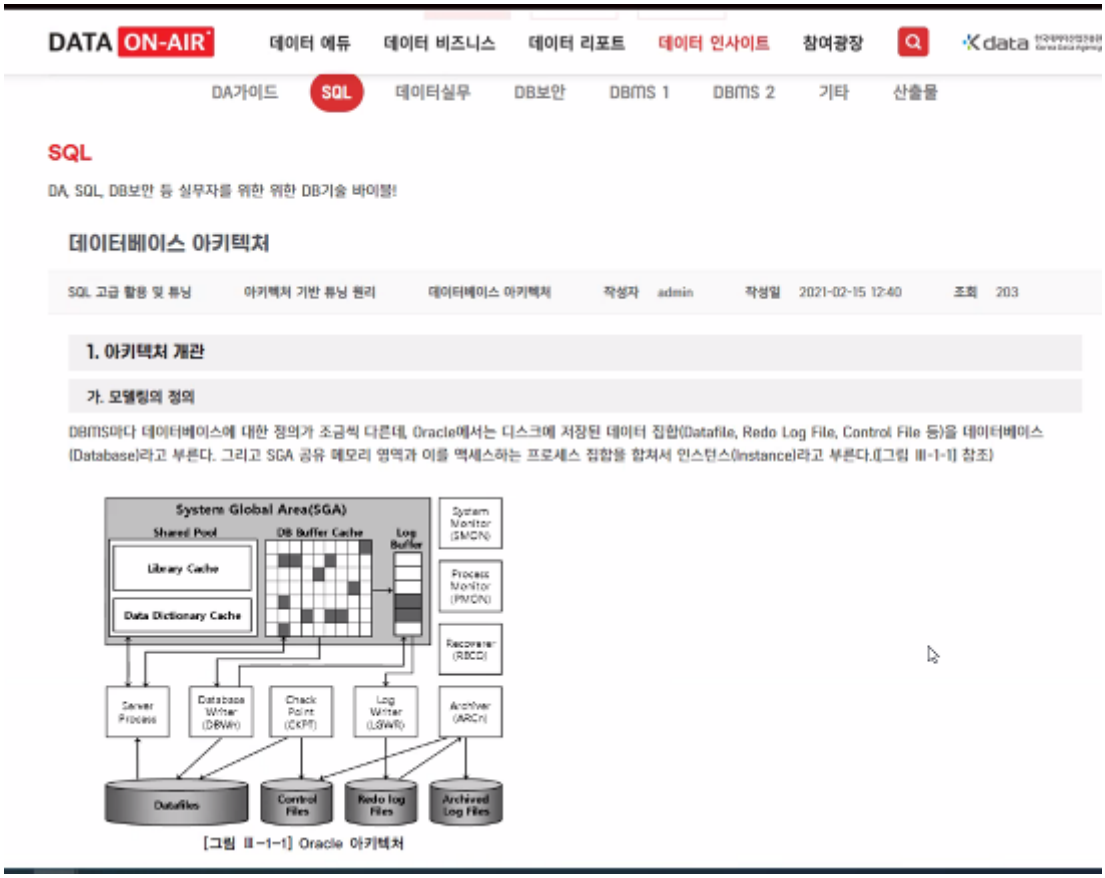
데이터 베이스 아키텍처

데이터 베이스의 전문가가 돼야 분석



백앤드 쪽에서 로직을 다 만든다. [알고리즘]

프론트 앤드에선 띄어주는 것.

근데 문제는 프론트앤드에서 하는 경우 많음, 현업에서도.

근데 문제는 백앤드가 탄탄하면

프론트앤드는 조금씩만 고치면 된다.

그런데 데이터베이스 저장만하고 프론트앤드만 로직을 쌓으면

그러면 뭔가 바꿀 때 엄청 많이 바꿔야 한다.

누더기가 된다.

전산시스템 잘 되는 곳 보면 백앤드가 탄탄하다.

실기시험 - 알고리즘 1~100 더하는 루프 사용해서 프로그램 만들어라.
C, Python으로 구현할 때 4~5줄.
과거에는 포트란으로 만들 때 30~40줄.

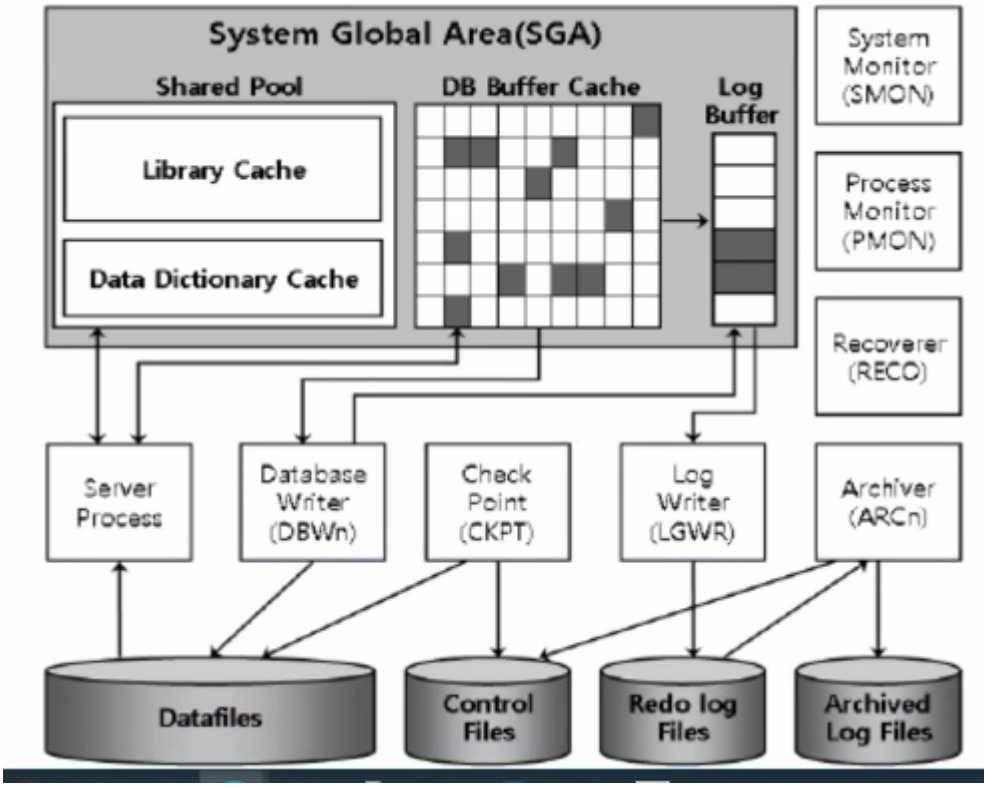
라이브러리 - 갖다 쓰면 된다. [함수, 스칼라- 외우는 거 아님]
우리가 필요한 건 통찰력, 인사이트.
[코드를 이해만 하면 된다.]

이미 회사에 아키텍처, 뼈 대가 다 되어 있다.
그 뼈대를 갖고 조금씩 하는 거지.
제로베이스에서 건물 짓는 거 아님. [원시시대]
[응용-빌드]
그럴 수도 없고, 그럴 시간도 없다.
다 만들어져 있기 때문에.

JAVA 스터디

프레임 = 스프링, 봄
아키텍처 다 되어 있다. 스프링 루트.

데이터베이스 아키텍처



RDBMS가 정확한 표현이다.

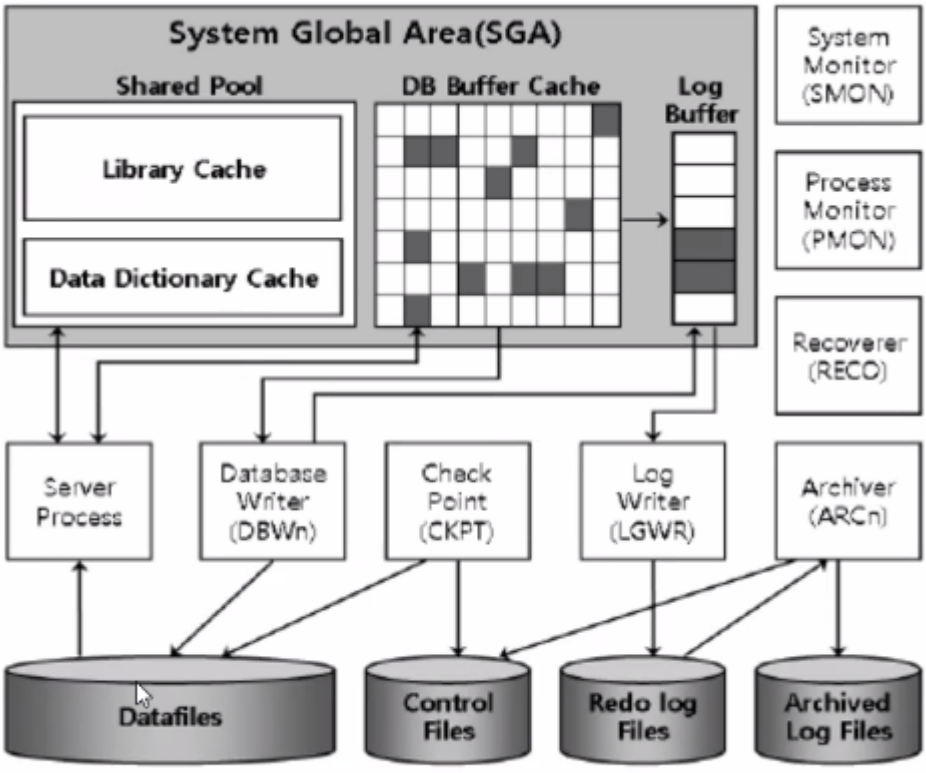
정의가 조금 이상하게 되어 있는 거 같다고 하심.

- 데이터 파일
  = HDFS, 수천개 테이블들이 저장 된 곳.
  오라클 아키텍처 그림[위]

- Redo log
  do 한다. 할 수 있다. 리두.
  do를 거꾸로 한다.
  RDBMS 보면 시스템 다운될 때, 전원이 확 나가면 시스템 망가질 수 있는데. 그런 장애를 예방하기 위해서 세이브[commit]하고 되돌리는 것[roll back]을 하는 작업을 위해 log를 남긴다. 쭈욱 남김.
  DM에 대해서
  select는 로그에 안남김. 왜? 테이블 조회한 건 왜 안남기지... ? 검색은 DML에 포함 안돼. 그냥 본건데.

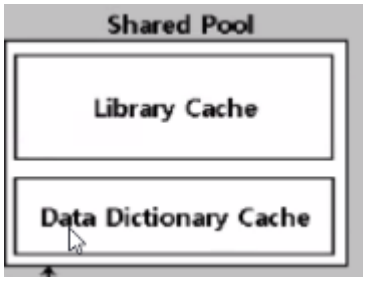
  단, 셀렉트 한 걸 보안상 저장할 수는 있다. 어느 작업했다는 걸 별도로 보관 가능.
  DML 에 대해서 Logging 하는 것 = Redo Log file.

  데이터 베이스 시험에 중요한 내용들. 로그라이터(백그라운드)



[그림 Ⅲ-1-1] Oracle 아키텍처

- DBWR, DBWn 저장하는 놈
- Check point 데이터 파일과 콘트롤 파일(작은놈)
  싱크로라이제이션(동기화 맞추기) 프로세스
  데이터 베이스 고장 대비한 것. 안정성 위한 싱크 맞추기
- 아카이브
  의미는? 건축? 데이터를 아카이빙한다.
  테이프는 말아서 저장하는 것. 백업 위해. 백업받는 파일이다.
- SGA **까만 부분** = 메모리, 수기가~수십기가~수백기가
  다른 건 몰라도 돼. 이것만 알면 돼
  하얀부분은 프로세스
  메모리에 하얀 건 시스템 관련 된 거(명령어, 관리 명령어)



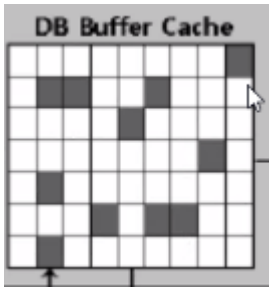
  세어프 폴은
  셀렉트 등 - 데이터베이스에 명령 내리면 알아먹지 못해.
  그걸 팔싱한다. 번역을 한다. RDBMS가 알아듣게금 = 파스한다. 팔싱한다.
  춘향이도 똑같이 신청. 그러면 파싱하지 않고 이전에 썼던 문장을 갖다 쓴다. 파싱 안하고. 아는 단어, 아는 뜻. 효율이 좋은 것.

  근데 조금 변경해서 날리면?
  RDBMS 입장에선 새로 파싱함.

  - 파싱된 결과가 library cache

  SQL 문장들을 파싱해서 저장하는 곳 = Shared Pool

- 덜티블랙



세이브 된 내용들/ 뭘 적을 때. 로그가 500개면 그게 검은색으로 들어감. 그걸 또 누가 똑같이 부르면 거기서 가져 감.

새로 하면 데이터파일로 가서 찾아서 갖고 와서 디비버퍼카케?

데이터 조회하면 그 데이터들이 이 메머리 버퍼 켓에 올라가는 것.

다 차면 다 반환해.

자주 사용 되면 (히트되면) 동일 데이터 참조 계속 하면 DB 입장에선 좋은 것. 그런데 디스크로 내려가는 순간. 읽고 내리고 하느 그런 걸 I/O라고 한다. 이 아이오때문에 많은 어려움을 겪곤 한다.

데이터베이스가 디스크 아이오 작업 없도록
메모리에 다 올리는 추세. 메모리 디비라고 한다.
인 메모리 디비를 많이 사용하는 추세이다. 비싸. 날라가면 싹 다 날라간다.

현업에선 데이터 베이스 구조 + 메모리 디비.

쉐어드 풀, 디비버퍼켓, 로그버퍼(디엠작업 저장-리두로그파일에 저장; 나중에 장애 방지위해, 로그 다 기억)

이거 잘 기억, 데이터 베이스 구조익히는 것 중요
요런 식으로 작동. 자주 사용되는 SQL 문장을 만들어야 겠다.

더 나가면 오래걸리니깐 참자.

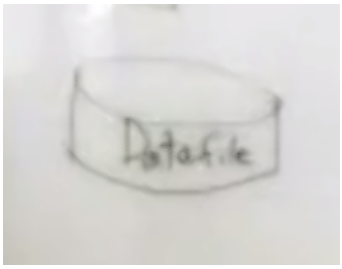
클러스터 RAS

클러스터 = 디바이스 끼리 연결 시킨 거다.
컴퓨터 끼리 묶은 걸 클러스터라고 한다.

인스턴스
사용자 그룹. 회사에서 데이터베이스가 한개라고 하면 여러 그룹이 공용사용

근데 보통 인사쪽이랑 자제쪽이랑 데이터베이스가 다르다.
2개의 인스턴스가 있다고 한다.
개별 별도의 부서별 데이터베이스라고 생각하면 된다. 인스턴스.

하나의 인스턴스. 여러 인스턴스 이
RAC = 디비버퍼 + 풀 + 로그



테이블 저장 된 곳

오라클 인스턴스라고 치면 / 2개 만드는 이유 뭐냐면

데이터파일이 제일 중요; 모든 데이터 테이블.

C storage

A랑 B는 물리적 다른 곳.

데이터 파일 바라보고 있다가
a가 죽으면 사용자 많을 텐데. 그 사용자들이 b로 이사. 그게 **RAC**

일을 못하니깐. 망가지는 순간 이주한다. 중단 없이 업무하는 걸 RAC

real application cluster
죽어도 상관없어-

이런 얘기 현업에서 많이 사용함.

요 개념들과 용어

오라클 - 80% 이상이 이걸로 되어있어서 오라클로 설명한다(대한민국 내)

디비 쪽에서 일하려면 오라클 접할 수밖에 없다.

디비 아키텍처 이렇게 생겨서
쿼리를 재사용할 수 있게 만들어야 겠다. 라고 생각하고 일하는 것.

셀렉트 * 프롬 이엔피 ;
이걸 띄어쓰기 다르고 하면 재사용 안된다. 같은 문법.
수십수백명이 데이터베이스 쓰기 때문에 공유의 개념으로 데이터베이스랑 테이블, 쿼리 만드는 것.

서버 다운되는 문제
대부분이 I/O문제.

데이터 베이스 아무리 성능 좋아도 쿼리문 하나 잘 못 만들면 다운 가능.

SQL 문장 쓸 때 신중을 기해야 한다.
그리고 점검하는 습관을 만들어야 한다.

만들고 땅 때리면
데이터베이스에 혹독한 지시를 주는 것. 2억건 곱하기해, 조인해, 그러면 할 수가 없다. 디비 죽는다.