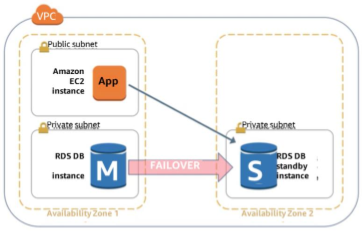
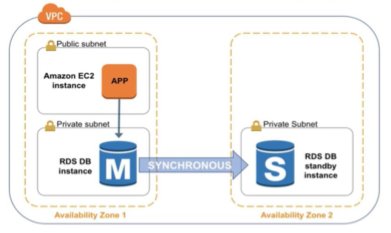
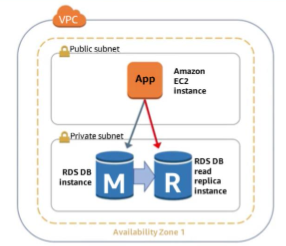
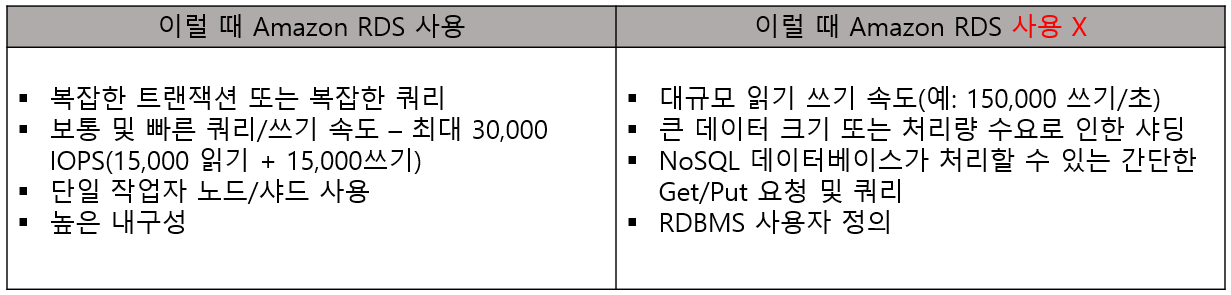
1. 파트 1: Amazon Relational Database Service (Amazon RDS)
   1. 비관리형과 관리형 서비스 비교
      1. 비관리형: 사용자가 확장하며 내결함성 및 가용성 관리
      2. 관리형: 일반적으로 확장하며 내결함성 및 가용성이 서비스에 내장
   2. 관계형 데이터베이스에는?
      1. RDB
      2. SQL
      3. structured(구조화된)
      4. 정형
      5. 트랜잭션 기반
      6. 복잡한 쿼리
      7. RSD
   3. 관계형 데이터베이스의 문제점
      1. 서버 유지 관리와 에너지 사용량
      2. 소프트웨어 설치 및 패치
      3. 데이터베이스 백업 및 고가용성
      4. 확장성의 제한
      5. 데이터 보안
      6. 운영체제 설치와 패치
   4. 관리형 서비스의 책임
      1. 고객이 관리
         * 애플리케이션 최적화
      2. AWS 관리
         * 운영체제 설치 및 패치
         * 데이터베이스 설치 및 패치
         * 데이터베이스 백업
         * 고가용성
         * 확장성
         * 전기와 랙, 적재
         * 서버 유지 관리
   5. Amazon RDS DB 인스턴스
      1. DB 인스턴스 클래스: CPU / 메모리 / 네트워크 성능
      2. DB 인스턴스 스토리지: 마그네틱 / 범용(SSD) / 프로비저닝된 IOPS
   6. 여러 가용 영역을 통한 고가용성(사진 출처: 강의 자료)

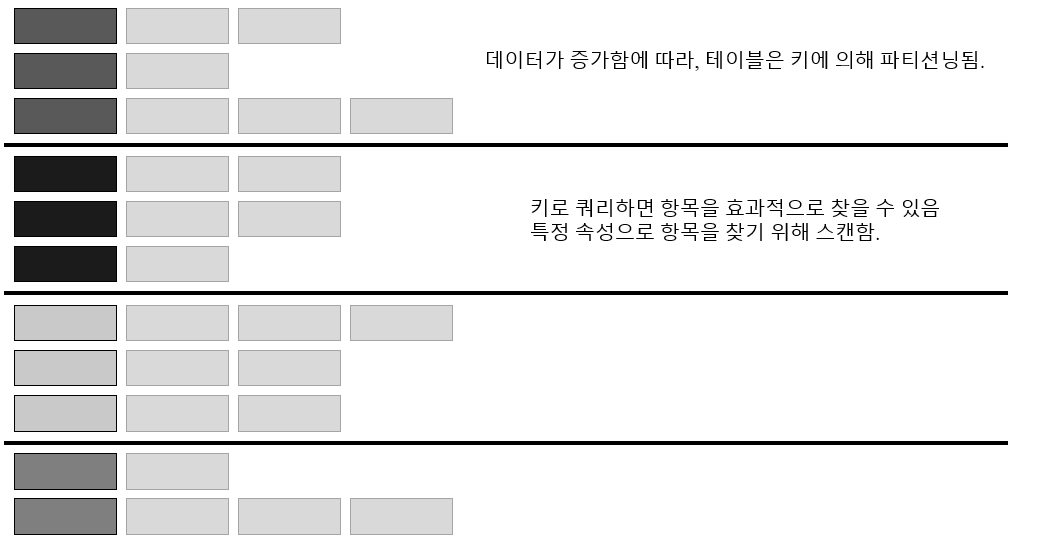


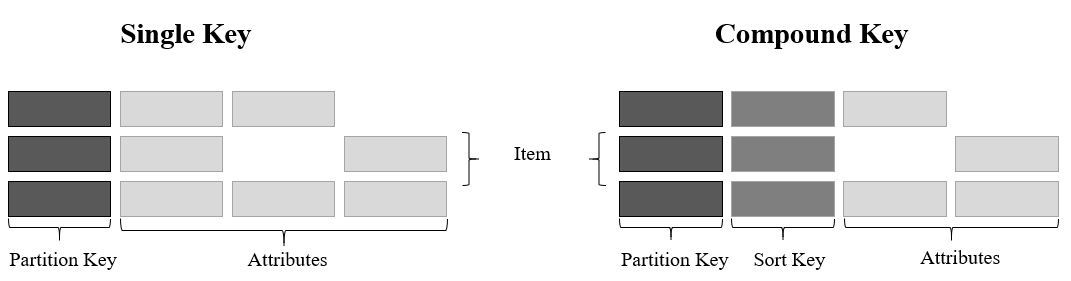
* 1. Amazon RDS 읽기 복제본(사진 출처: 강의 자료)
     1. 특징
        + 비동기식 복제
        + 필요할 때 마스터로 승격
     2. 기능성
        + 읽기가 많은 데이터베이스 워크로드
        + 읽기 쿼리 부하 분리
  2. Amazon RDS
     1. 사용 시기



* + 1. 매시간 청구
       - 리소스가 운영될 때 청구 개시
    2. 데이터베이스 특징
       - 데이터베이스의 물리적 용량: 엔진 / 사이즈 / 메모리 클래스
    3. DB 구매 유형
       - 온디맨드 데이터베이스 인스턴스: 시간당 컴퓨팅 용량
       - 예약 데이터베이스 인스턴스: 1년 또는 3년 기간의 낮은 가격, 일회성, 선지불형으로 데이터베이스 인스턴스를 예약
    4. DB 인스턴스 수
       - 최대 부하에도 운영되게 하기 위해 여러 DB 인스턴스를 프로비저닝함.
    5. 스토리지
       - 프로비저닝된 스토리지
         * 비용 없음 🡺 실행중인 데이터베이스의 백업을 위한 스토리지  
            (데이터베이스 스토리지의 100%까지)
         * 청구(GB/월) 🡺 종료된 DB인스턴스의 백업 스토리지
       - 추가 스토리지
         * 추가적으로 프로지버닝된 백업 스토리지
    6. 요청
       - 데이터베이스에 이루어지는 input/output 요청 수
    7. 배포 유형
       - 단일 가용 영역과 여러 가용 영역에 따라서 스토리지와 I/O 비용 차이 있음.
    8. 데이터 전송
       - 내부로 들어오는 데이터 전송에는 비용 없음.
       - 외부로 나가는 데이터 전송에는 계층적 요금 청구
  1. 요약  
     🡪 클라우드에서 관계형 데이터베이스 설치하고, 운영하고 확장하는데 특징은?
     1. 관리형 서비스
     2. 콘솔, AWS RDS CLI 또는 단순 API calls로 접근 가능
     3. 확장성 있는 compute와 스토리지
     4. 자동화된 중복과 백업 가능
     5. 지원되는 데이터베이스 엔진
        + Amazon Aurora, PostgreSQL, MySQL, MariaDB, QRACLE, Microsoft SQL Server

1. 파트 2: Amazon DynamoDB
   1. Amazon DynamoDB란 무엇입니까?
      1. 비관계형 데이터베이스이며 비정형이고, 키 값에 패턴을 가짐.
      2. NoSQL 데이터베이스 서비스임.
      3. NoSQL 데이터베이스 테이블
      4. 무한 스토리지(가상적)
      5. 항목은 속성을 달리할 수 있음.
      6. 낮은 지연시간의 쿼리
      7. 확장성 있는 읽기/쓰기 처리량
   2. 핵심 구성 요소
      1. 테이블
      2. 항목
      3. 속성
      4. 다른 두 종류의 기본 키 지원: 파티션 키 / 파티션 및 정렬 키
   3. 파티셔닝



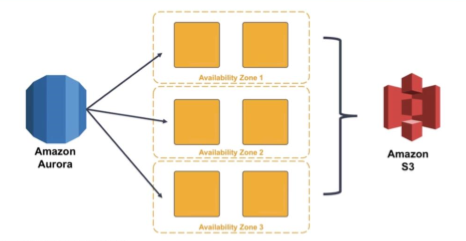


* 1. 검토
     1. SSD에서만 실행됨.
     2. 문서 및 키 값 스토어 모델 지원
     3. AWS 리전 선택에 의해 자동적으로 DynamoDB테이블 복제되는 글로벌 테이블 기능을 가짐.
     4. 모바일, 웹, 게임, 광고 기술 및 IoT애플리케이션에 적합함.
     5. AWS Management Console, AWS 명령줄 인터페이스 또는 간단한 API 호출을 통해 액세스 가능함.
  2. **요약(\*시험에 잘 나옴\*)**
     1. **DynamoDB는 완전 관리형 NoSQL 데이터베이스 서비스임.**
     2. **규모와 관계없이 일관되게 10밀리초 미만의 지연시간 제공**
     3. **테이블 크기 및 처리량 제한 없음**
     4. **글로벌 테이블은 리전 간 데이터 복제와 업데이트 충돌 해결의 어려움을 없앰.**

1. 파트 3: Amazon Redshift
   1. 소개
      1. 빠르고 강력하며 완전 관리형 데이터웨어하우스
      2. 설정하고 사용하고 확장하는데 비용이 효과적이면서 간단
      3. 정교한 쿼리 최적화, 고성능 로컬 디스크의 컬럼형 스토리지 및 대규모 **병렬 쿼리 실행** 사용 🡪 petabyte 단위의 구조화 데이터에 대해 복잡한 분석 쿼리 시행 가능. (대부분의 결과는 수초만에 나타남.)
   2. 사용 사례
      1. 엔터프라이즈 데이터웨어하우스(EDW)
         * 고객이 편안하게 사용할 수 있는 속도로 이전
         * 많은 선지물이나 약정 없이 시험 가능
         * 비즈니스 요구에 빠른 대응
      2. 빅데이터
         * 소규모 고객을 위한 낮은 가격대
         * 배포와 유지관리가 쉬운 관리형 서비스
         * 데이터베이스 운영업무를 줄이고, 데이터에 더욱 집중할 수 있도록 함.
      3. Software as a Service(SaaS)
         * 수요가 증가함에 따라 데이터웨어하우스 용량도 확장
         * 애플리케이션에 분석기능 추가
         * 규모에 따라 하드웨어와 소프트웨어 비용 절감
   3. 요약
      1. 빠른, 완전 관리형 데이터 웨어하우스 서비스
      2. 중지하지 않고 손쉽게 확장
      3. 컬럼 스토리지와 병렬 프로세싱 아키텍처
      4. 자동적이고 지속적으로 클러스터 모니터링
      5. 암호화 내장
2. 파트 4: Amazon Aurora
   1. 소개
      1. 클라우드용으로 구축된 MySQL 및 PostgreSQL 호환 관계형 데이터베이스
      2. 고성능 상용 데이터베이스의 성능 및 가용성과 오픈 소스 데이터베이스의 단순성 및 비용 효율성 결합
   2. 서비스 혜택(사진 출처: 강의 자료)



* 1. 고가용성(사진 출처: 강의 자료)



* 1. 요약
     1. 고성능과 확장성
     2. 고가용성과 내구성
     3. 다 계층 보안
     4. MySQL과 PostgreSQL 호환
     5. 완전 관리형 서비스
     6. 크기 🡺 64TiB  
        기존 DB(MySQL, MSSQL, ORACLE)은 16TiB  
        왜냐하면 EBS가 최대 16TiB 이기 때문임.