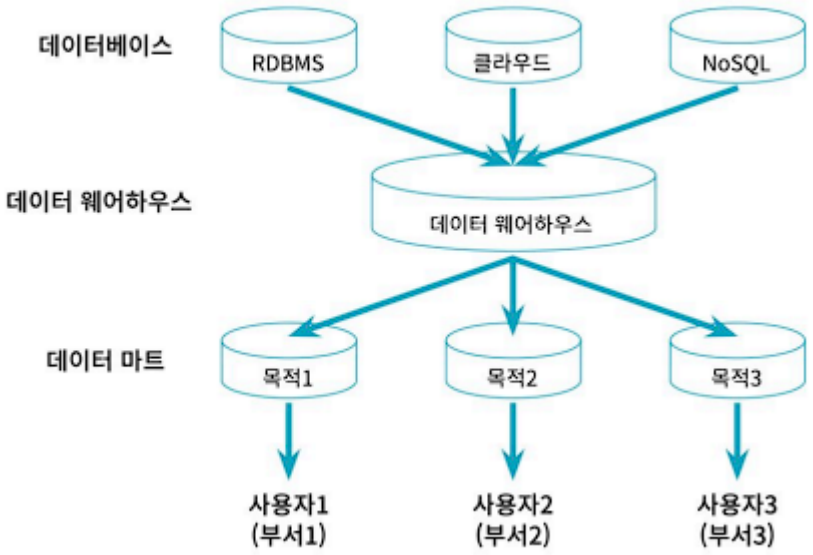
02. 데이터 마트

1. 데이터 마트의 이해

- **데이터 마트**
  - 데이터 분석을 하기에 앞서 분석 목적에 맞춰 데이터를 수집, 변형하는 과정이 필요, 데이터 웨어하우스로부터 특정 사용자가 관심을 갖는 데이터들을 주제별, 부서별로 추출하여 모은 비교적 작은 규모의 데이터 웨어하우스
  - 데이터 마트 개발 : 위처럼 데이터를 수집하고 변형하여 한 곳에 모으는 작업

【 데이터 웨어하우스와 데이터 마트 】



- 데이터 전처리
  - 데이트 마트 개발 후 빅데이터 분석 단계 전 데이터를 전처리(preprocessing)하는 과정이 필요
  - 데이터 정제(Cleansing), 분석 변수 처리 과정이 포함
    - 데이터 정제 : 결측값과 이상값을 처리
    - 분석 변수 처리 과정 : 변수 선택, 차원 축소, 파생변수 생성, 변수 변환, 클래스 불균형(불균형 데이터 처리) 등
  - 요약변수와 파생변수
    - 요약변수 : 원래의 데이터로부터 기본적인 통계 자료를 추출한 변수
    - 파생변수 : 특정 목적을 갖고 조건을 만족하는 변수를 새롭게 생성한 것

2. 데이터 마트 개발을 위한 R패키지 활용

- reshape 패키지
  - melt : 데이터를 특정 변수를 기준으로 녹여서 나머지 변수에 대한 세분화된 데이터를 생성

```
> score # 두 학생의 학기별 점수 데이터프레임
  student_number   semester math_score english_score
1              1          1         60            80
2              2          1         90            70
3              1          2         70            40
4              2          2         90            60

# reshape가 설치 및 호출되지 않을 경우 reshape2를 설치해도 된다.
> library( reshape )
> melt( score , id = c( "student_number" , "semester" ) )
  student_number   semester        variable   value
1              1          1     math_score      60
2              2          1     math_score      90
3              1          2     math_score      70
4              2          2     math_score      90
5              1          1   english_score      80
6              2          1   english_score      70
7              1          2   english_score      40
8              2          2   english_score      60
```

  - cast : melt에 의해 녹은 데이터를 요약을 위해 새롭게 가공 할 수 있도록 도움

```
# melt를 활용하여 얻은 결과를 melted_score로 저장
> melted_score <- melt( score , id = c( "student_number" , "semester" ) )
>
# 학생의 과목별 평균점수를 알고 싶은 경우
> cast( melted_score , student_number ~ variable , mean )
  student_number  math_score  english_score
1              1          65             60
2              2          90             65
>
# 학생의 학기별 평균점수를 알고 싶은 경우
> cast( melted_score , student_number ~ semester , mean )
  student_number   1   2
1              1  70  55
2              2  80  75
>
# 학생의 과목별 최댓값을 알고 싶은 경우
> cast( melted_score , student_number ~ variable , max )
  student_number  math_score  english_score
1              1          70             80
2              2          90             70
```

- sqldf 패키지
  - 표준 SQL 문장을 활용하여 R에서 데이터프레임을 다루는 것을 가능하게 해주는 패키지(SAS에서 PROC SQL과 같은 역할)

```
> library( sqldf )
> sqldf( 'select * from score' )
  student_number   semester  math_score  english_score
1              1          1          60             80
2              2          1          90             70
3              1          2          70             40
4              2          2          90             60
>
> sqldf( 'select * from score where student_number = 1' )
  student_number  semester  math_score  english_score
1              1         1          60             80
2              1         2          70             40
>
> sqldf( 'select avg(math_score) , avg(english_socre) from score group by student_number' )
  student_number  avg(math_score)  avg(english_score)
1              1              65                  65
2              2              85                  65
```

- plyr 패키지

- apply함수를 기반으로 데이터를 분리하고 다시 결합하는 가장 필수적인 데이터 처리 기능을 제공
- 입력되는 데이터 구조와 출력되는 데이터 구조에 따라 여러 가지 수를 지원

		입력 데이터 구조		
		데이터프레임	리스트	배열
출력 데이터 구조	데이터프레임	ddply	ldply	adply
	리스트	dlply	llply	alply
	배열	daply	laply	aaply

- data.table 패키지
  - 특정 칼럼별로 주솟값을 갖는 인덱스를 생성하여 연산 및 검색을 빠르게 수행할 수 있는 데이터 구조

```
> year <- rep( c( 2012:2015 ) , each = 12000000 )
> month <- rep( rep( c( 1:12 ) , each = 1000000 ) , 4 )
> value <- runif( 48000000 )
# 같은 데이터로 4800만 개의 행을 갖는 데이터프레임과 데이터 테이블을 생성
> DataFrame <- data.frame( year , month , value )
> DataTable <- as.data.table( DataFrame )
# 데이터프레임의 검색 시간을 측정
> system.time( DataFrame[ DataFrame$year == 2012 , ] )
```

03. 데이터 탐색

1. 탐색적 데이터 분석(Exploratory Data Analysis, EDA)
   - 데이터를 이해하고 의미 있는 관계를 찾아내기 위해 데이터의 통곗값과 분포 등을 시각화하고 분석하는 것
2. 결측값
   - 결측값 : 존재하지 않는 데이터
     - NA(Not Available)로 표현, null, 공백, -1 등 다양하게 표현
     - 대표적인 패키지 : Amelia와 DMwR2패키지

참고 Amelia 패키지에 내장되어 있는 missmap 함수로 결측값 시각화

- Amelia 패키지에 내장되어 있는 missmap은 데이터프레임에서 결측값이 어디에 존재하고 어떻게 분포되어 있는지 한눈에 시각화하여 사용자에게 보여준다.

```
> copy_iris <- iris # 원본 데이터를 유지
> copy_iris[ sample( 1 : 150 , 30 ) , 1 ] <- NA # Sepal.Length에 30개의 결측값 생성
> library( Amelia ) # Amelia 패키지가 없다면 install.packages( "Amelia" )를 우선 실행
> missmap( copy_iris )
```

- **결측값 대치 방법**
  - 단순 대치법
    - 결측값이 존재하는 데이터를 삭제하는 방법
    - 대량의 데이터 손실 발생 가능
    - complete.cases함수 : 하나의 열에 결측값이 존재하면 FALSE, 존재하지 않으면 TRUE 반환
  - 평균 대치법
    - 평균 혹은 중앙값으로 결측값을 대치하여 불완전한 자료를 완전한 자료로 만드는 방법
    - 비조건부 평균 대치법 : 데이터의 평균값으로 결측값을 대치
    - 조건부 평균 대치법 : 실제 값들을 분석하여 회귀분석을 활용하는 대치 방법
    - DMwR2 패키지의 central Imputtation함수
  - 단순 확률 대치법
    - 평균 대치법에서 추정량 표준 오차의 과소 추정 문제를 보완하고자 고안된 방법
    - K-Nearest Neighbor 방법
      - K 최근접 이웃 알고리즘으로 주변 K개의 데이터 중 가장 많은 데이터로 대치하는 방법
      - 주변 몇개의 데이터가 결측값을 대치하기 좋은가에 대한 K를 선정하기가 쉽지 않음
  - 다중 대치법
    - 여러 번의 대치를 통해 n개의 임의 완전자료를 만드는 방법
    - **결측값 대치, 분석, 결합**의 세 단계로 구성

3. 이상값

- 이상값 : 다른 데이터와 비교하였을 때 극단적으로 크거나 극단적으로 작은 값
- 이상값 판단
  - ESD(Extreme Studentized Deviation)
    - 평균으로부터 '표준편차 3'만큼 떨어진 값들을 이상값으로 인식하는 방법
  - 사분위수
    - 사분위수를 이용하여 25%에 해당하는 값(Q1)과 75%에 해당하는 값(Q3)을 활용하여 이상치를 반단하는 방법
    - 일반적으로 사분범위에서 1.5분위수를 벗어나는 경우 이상치로 판단, Q1-1.5*IQR(하한 최솟값)보다 작거나 Q3+1.5*IQR(상한 최댓값)보다 큰 값은 이상값으로 간주

【 사분위수 】

```
# 테스트를 위한 임의 데이터 생성
> data <- c( 3 , 10 , 13 , 16 , 11 , 20 , 17 , 25 , 43 )
> boxplot( data , horizontal = T)
```