



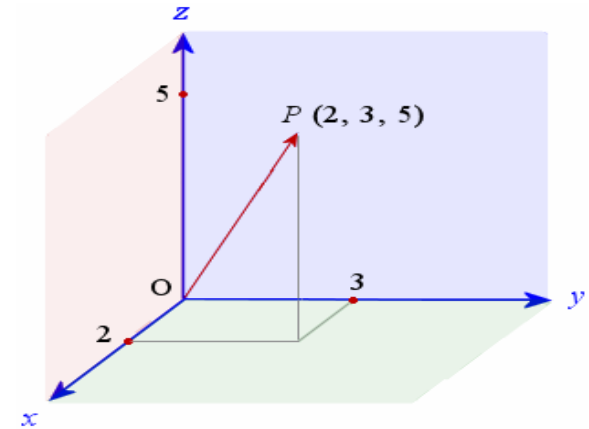
소프트웨어시스템 실습

6강: 다차원 데이터 구성 및 OLAP

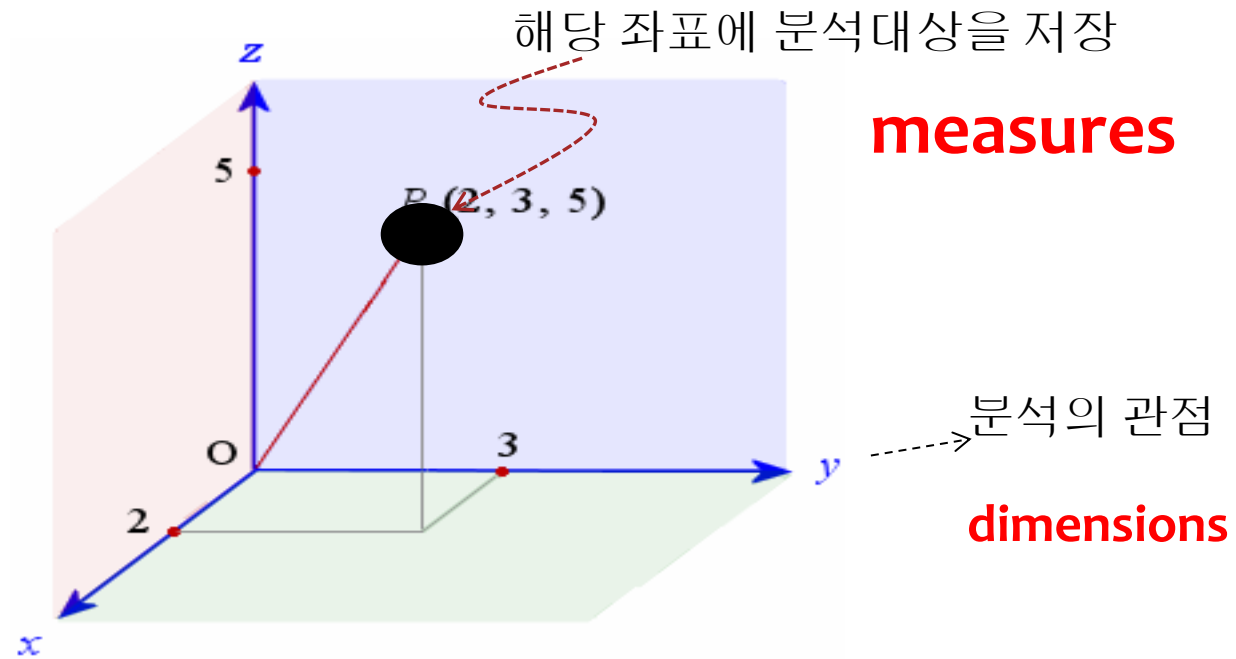


Multi-Dimensional Data

- ▶ Decision Making을 위해 특별히 설계된 'subject' 중심적인 데이터 저장소로서 다양한 'view'에서 관찰 분석이 가능
 - ▶ subject (measure) -> 'fact' 테이블
 - ▶ view -> 'dimension' 테이블

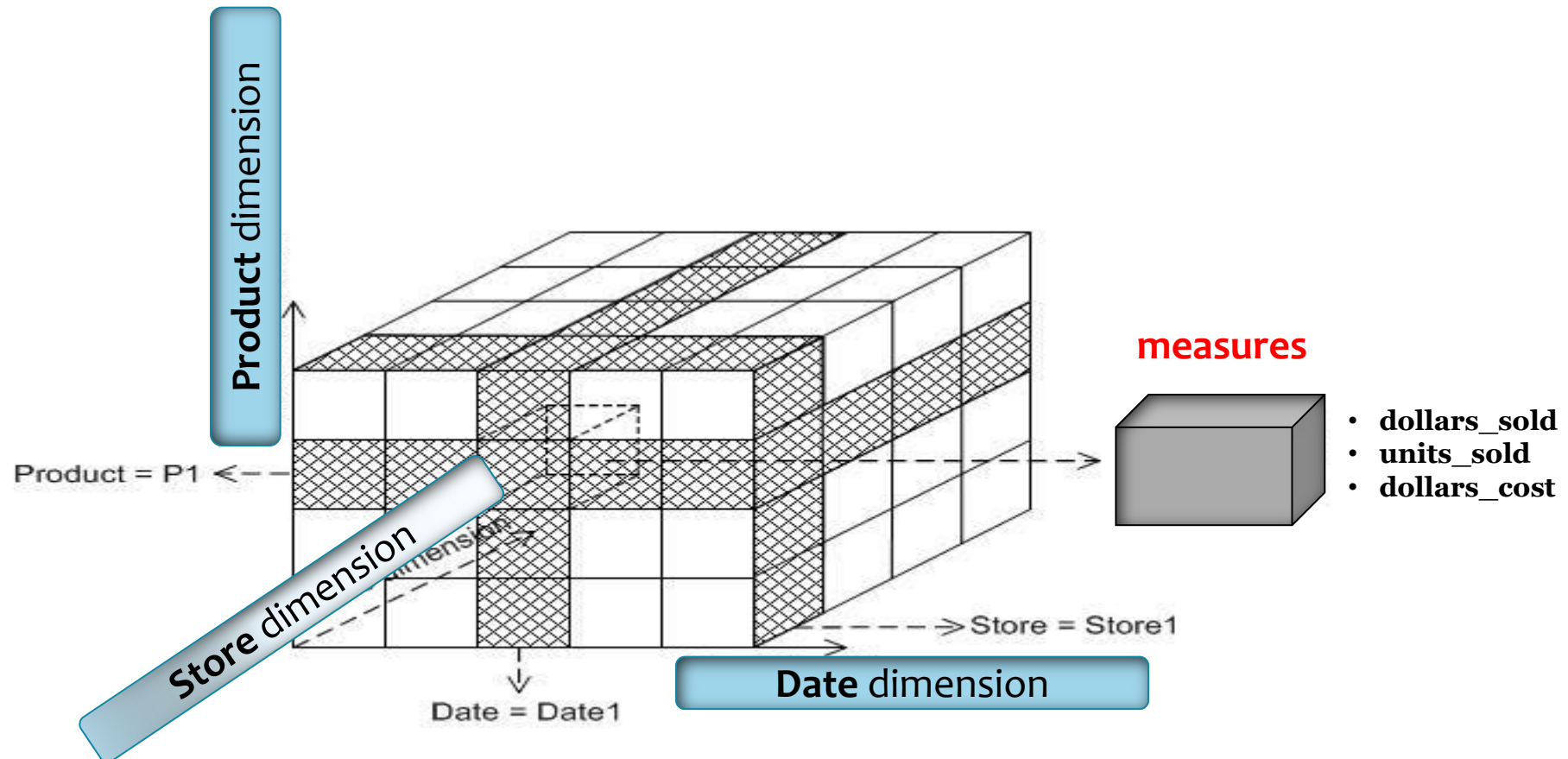


분석을 위한 DB의 구성 ?



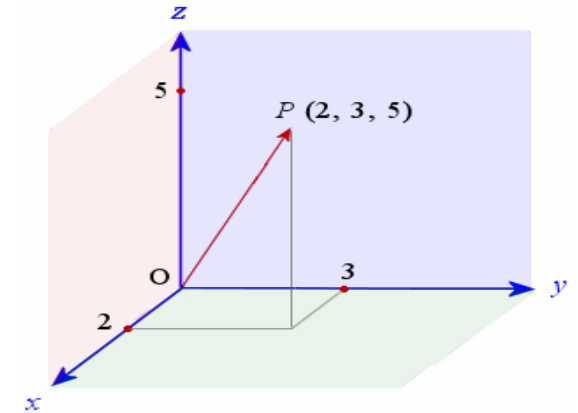
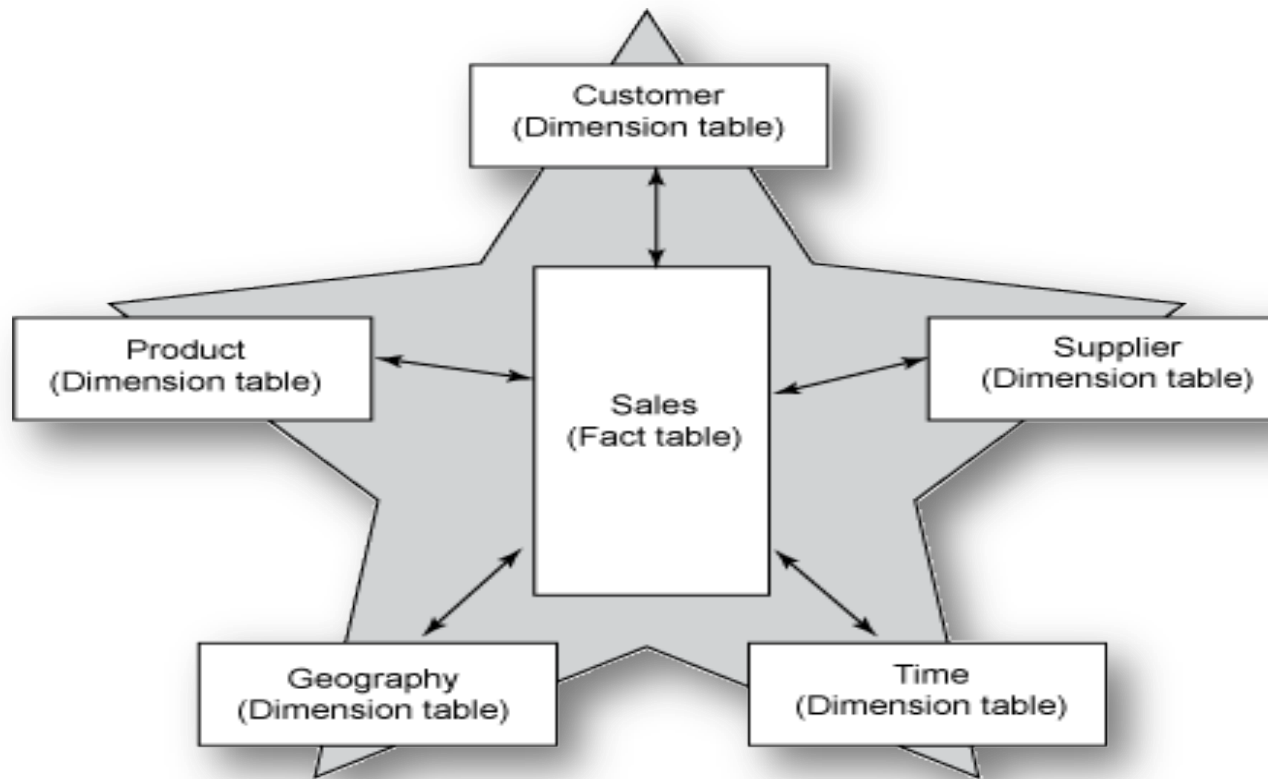
Multi-Dimensional DB 구축

▶ Multi-dimensional DB (n차원 배열형태) 로 구축



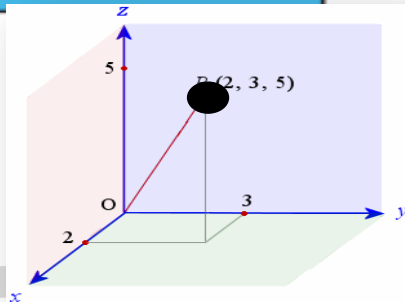
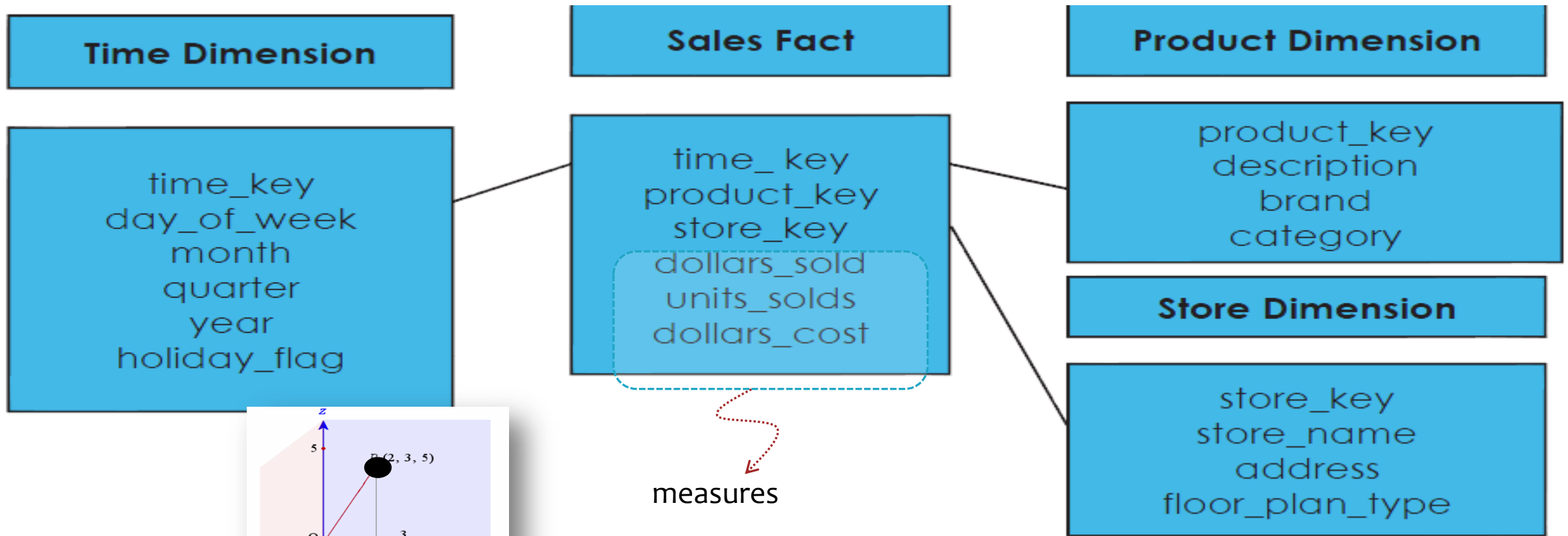
Multi-Dimensional DB 구축

▶ 관계형 데이터베이스에서 MDB(Data Warehouse)의 구축



Multi-Dimensional DB 스키마

- 관계형 데이터베이스에서 DW의 구축 : **Star schema model**



Multi-Dimensional DB 구축

▶ Relational DB로 구축

'Time' Dimension 테이블

time_key	day_of_week	month	quarter	year	holiday_flag

'Product' Dimension 테이블

'Store' Dimension 테이블

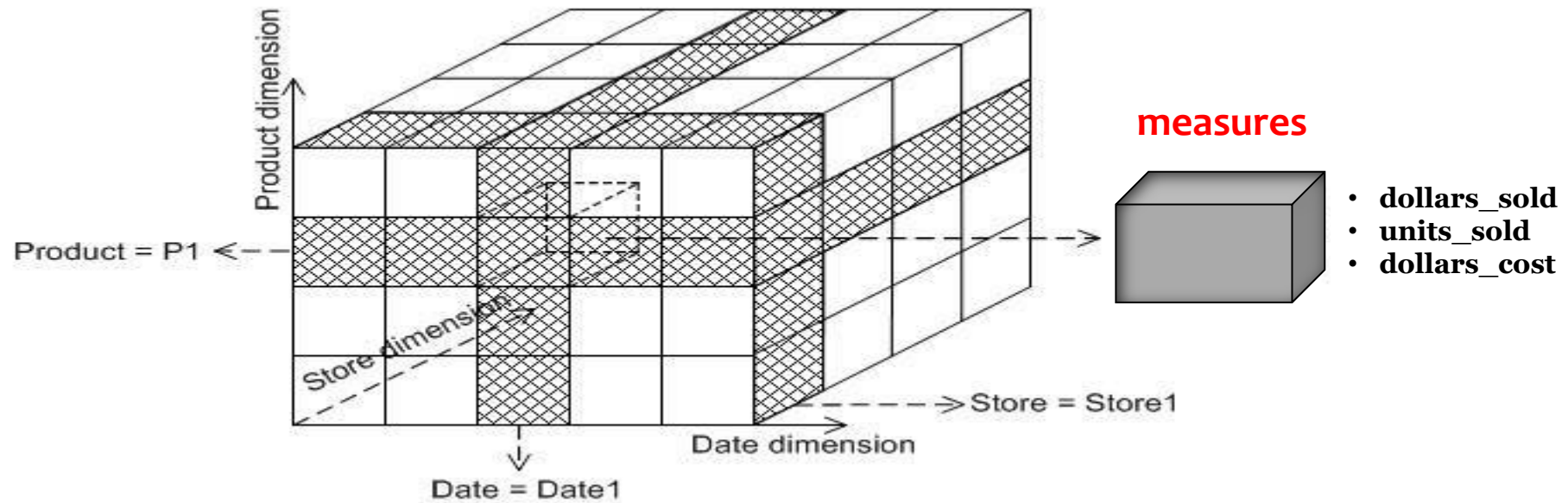
measures

time_key (FK)	product_key (FK)	store_key (FK)	dollars_sold	units_sold	dollars_cost

'Sales' Fact 테이블

Multi-Dimensional DB 구축

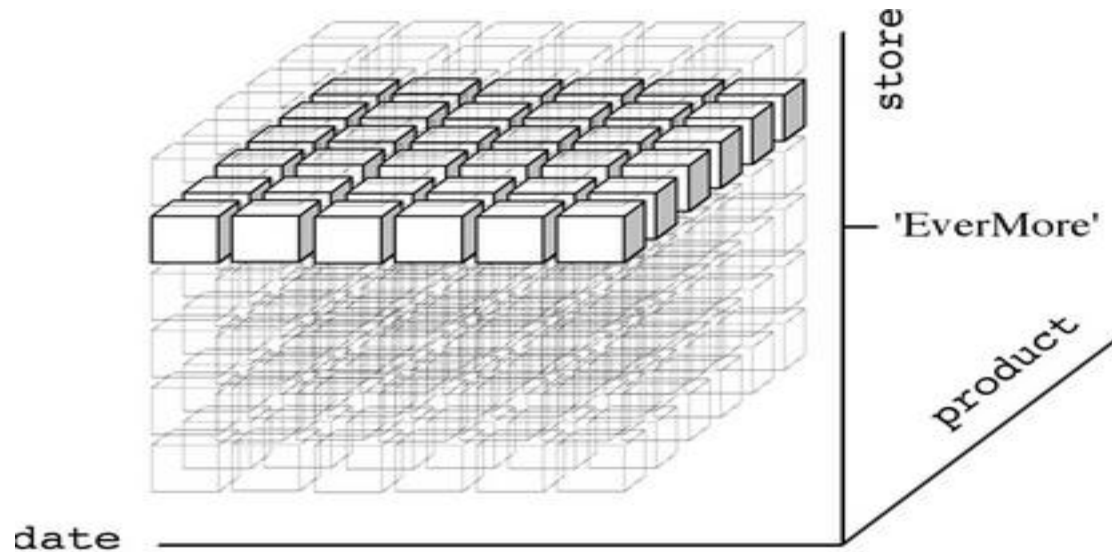
▶ Multi-dimensional DB로 구축



Multi-Dimensional DB 활용

▶ 연산: Slicing

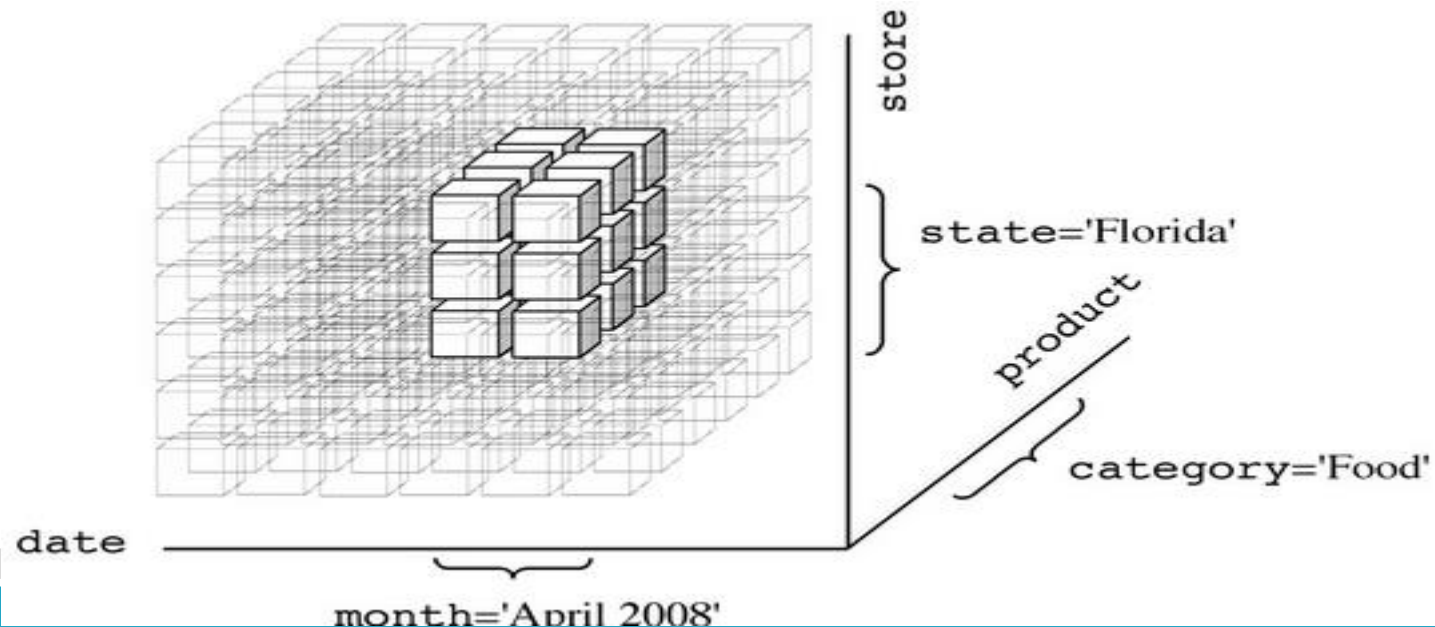
- ▶ 특정 차원을 고정시켜놓고 다른 차원들의 데이터를 관찰
- ▶ 예:
 - ▶ 'EverMore' 지점을 고정하여, date, product 차원에서 판매 현황 관찰



Multi-Dimensional DB 활용

▶ 연산: Dicing

- ▶ 각 차원마다 범위를 설정하여 보다 제한된 cube를 생성
- ▶ 예:
 - ▶ Time 차원 범위: 2008년 4월
 - ▶ Product 차원 범위: Food 카테고리
 - ▶ Store 차원 범위: Florida



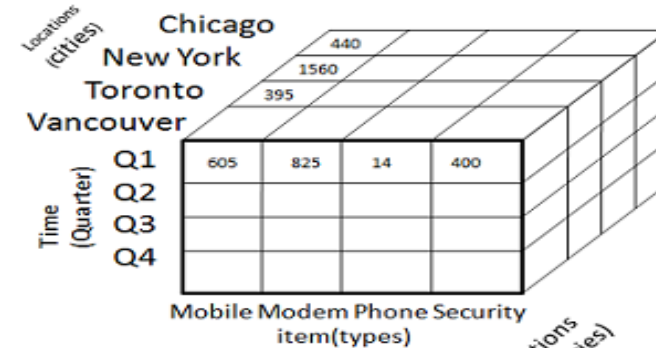
Multi-Dimensional DB 활용

▶ 연산: Drill-down ("Show me more detail")

- ▶ 상위 수준의 요약정보로부터 시작하여 단계적으로 관련된 구체 데이터를 추적하는 과정

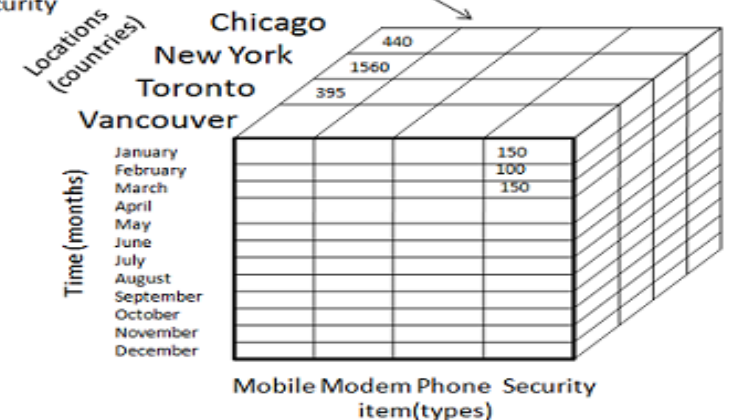
▶ 예:

- ▶ 2014년 분기별 판매량 정보
- ▶ 2014년 월별 판매량 정보



▶ 연산: Roll-up

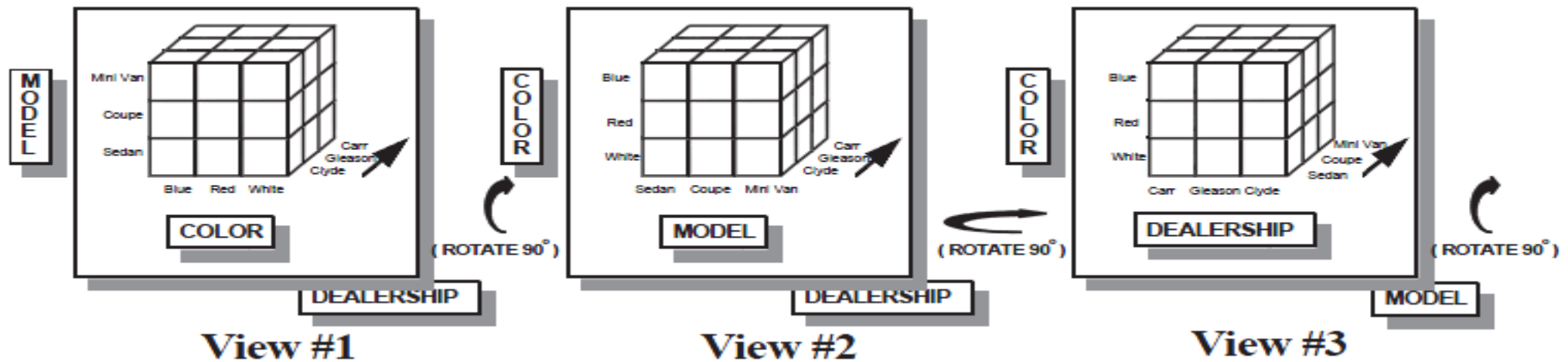
- ▶ drill-down의 반대 연산



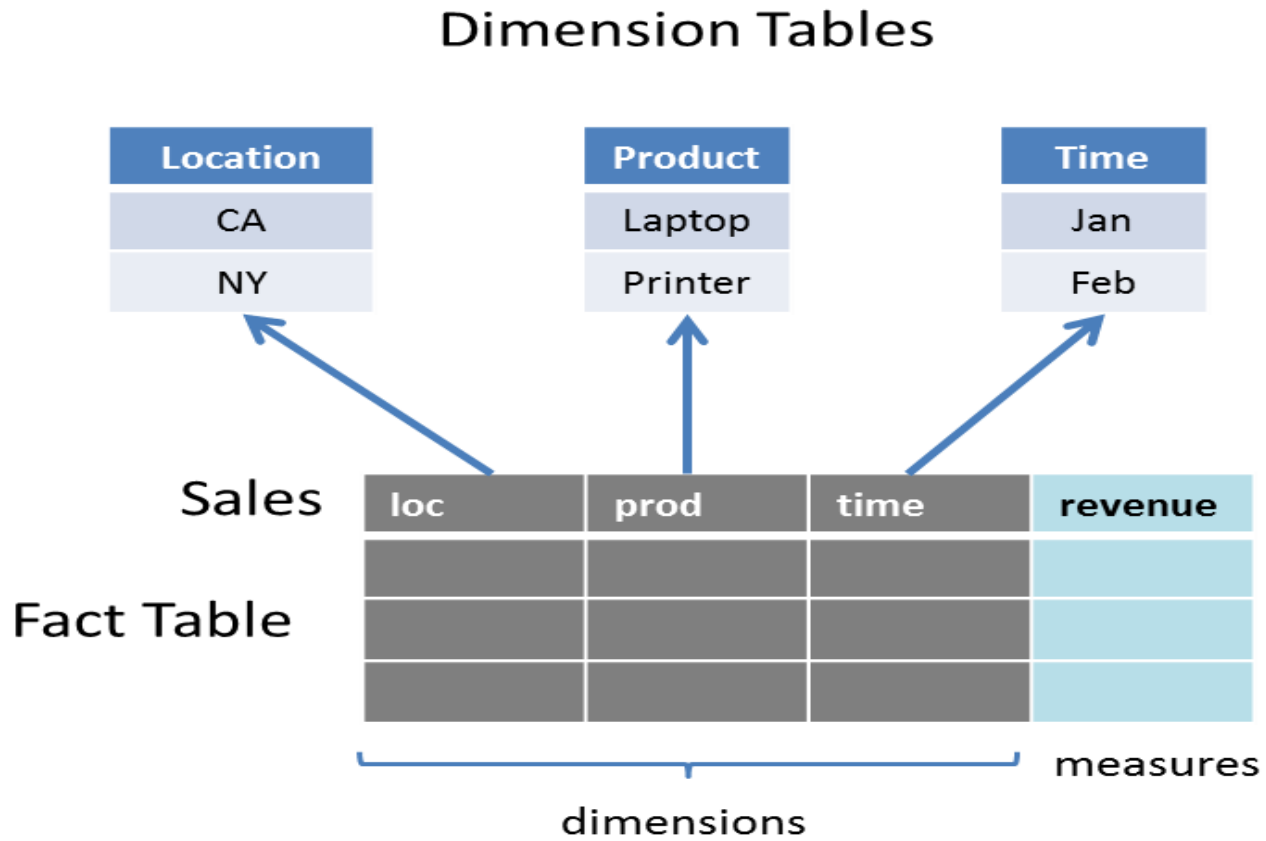
Multi-Dimensional DB 활용

▶ 연산: Pivoting (or Rotating)

- ▶ n개 차원의 방향을 전환
- ▶ 새로운 차원의 삽입하면서 수행할 수도 있음



Multi-Dimensional DB 스키마



Multi-Dimensional DB 구축: R의 활용

▶ Dimension 테이블의 구축

- ▶ **product** 차원 테이블: prod_table
- ▶ **time** 차원 테이블: month_table
- ▶ **location** 차원 테이블: state_table



```
state_table <- data.frame(key=c("CA", "NY", "WA", "ON", "QU"),  
  name=c("California", "new York", "Washington", "Ontario", "Quebec"),  
  country=c("USA", "USA", "USA", "Canada", "Canada"))
```

```
month_table <- data.frame(key=1:12,  
  desc=c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"),  
  quarter=c("Q1", "Q1", "Q1", "Q2", "Q2", "Q2", "Q3", "Q3", "Q3", "Q4", "Q4", "Q4"))
```

```
prod_table <- data.frame(key=c("Printer", "Tablet", "Laptop"),  
  price=c(225, 570, 1120))
```

Multi-Dimensional DB 구축: R의 활용

▶ Fact 테이블의 구축: random하게 measure 데이터 생성

```
gen_sales <- function(rec_num) {  
  # Generate 'fact' data randomly  
  loc <- sample(state_table$key, rec_num, replace=T, prob=c(2,2,1,1,1))
```

loc <- sample(state_table\$key, rec_num, replace=T, prob=c(2,2,1,1,1))

샘플링 대상

샘플링 횟수

중복허용

샘플링 확률

```
unit <- sample(c(1,2), rec_num, replace=T, prob=c(10, 3)) # 판매 개수  
amount <- unit * prod_table[prod,]$price # 판매금액 = 개수 * 단가
```

```
sales <- data.frame(month=time_month, year=time_year, loc=loc, prod=prod,  
                   unit=unit, amount=amount)  
sales <- sales[order(sales$year, sales$month),] # 시간에 따라 fact data의 sorting  
row.names(sales) <- NULL  
return(sales)  
}
```

○○○ vector 또는 data.frame의 정렬 ○○○

```
sales <- sales[order(sales$year, sales$month), ]
```

1차 정렬

2차 정렬

```
x <- c(10, 50, 40, 30, 5)  
order(x) # 5 1 4 3 2 를 출력
```

5번째 값 < 1번째 값 < 4번째 값 < 3번째 값 < 2번째 값

```
x <- x[c(5,1,4,3,2)] # => x <- x[order(x)]
```


Multi-Dimensional DB 구축: R의 활용

▶ 'Sales' Fact 테이블의 생성

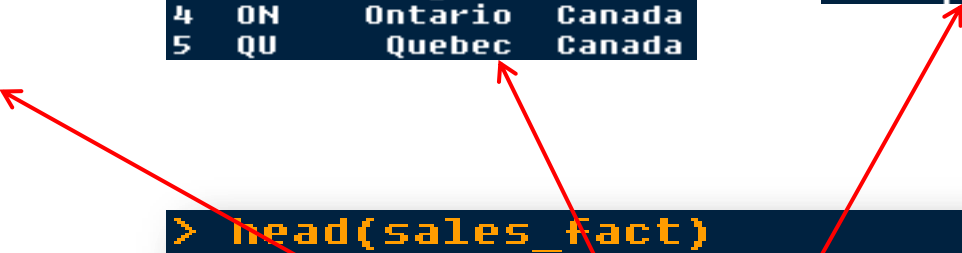
```
sales_fact <- gen_sales(500)
head(sales_fact)
```

```
> month_table
  key desc quarter
1    1  Jan     Q1
2    2  Feb     Q1
3    3  Mar     Q1
4    4  Apr     Q2
5    5  May     Q2
6    6  Jun     Q2
7    7  Jul     Q3
8    8  Aug     Q3
9    9  Sep     Q3
10   10 Oct     Q4
11   11 Nov     Q4
12   12 Dec     Q4
```

```
> state_table
  key name country
1  CA California USA
2  NY new York  USA
3  WA Washington USA
4  ON Ontario  Canada
5  QU Quebec   Canada
```

```
> prod_table
  key price
1 Printer  225
2 Tablet   570
3 Laptop  1120
```

```
> head(sales_fact)
  month year loc   prod unit amount
1     1  2012 ON Laptop    1    225
2     1  2012 NY Printer   2   1140
3     1  2012 ON Tablet    1   1120
4     1  2012 CA Tablet    1   1120
5     1  2012 QU Laptop    1    225
6     1  2012 CA Tablet    1   1120
```



Multi-Dimensional DB 구축: R의 활용

```
library(sqldf)
```

```
sqldf("select * from state_table")
```

	key	name	country
1	CA	California	USA
2	NY	new York	USA
3	WA	Washington	USA
4	ON	Ontario	Canada
5	QU	Quebec	Canada

표준 SQL 문장 처리

```
sqldf("select s.key, sum(amount)  
      from sales_fact f, state_table s  
      where f.loc=s.key  
      group by s.key ")
```

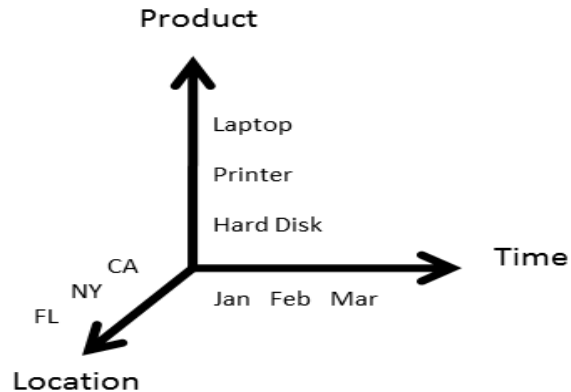
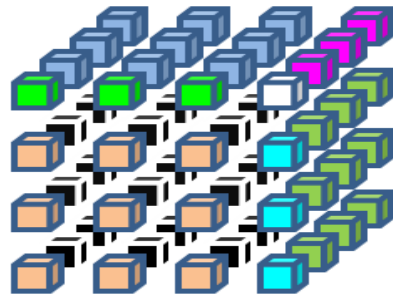
	key	sum (amount)
1	CA	137050
2	NY	143475
3	ON	57780
4	QU	47420
5	WA	53335

Multi-Dimensional DB 구축: R의 활용


Sales Fact Table

state	product	year	month	unit	amount




Hypercube






3D Cuboid

 Revenue of (Laptop, NY, Feb)


2D Cuboids

-  Revenue sum of (Laptop, Feb) over (locations)
-  Revenue sum of (NY, Feb) over (Products)
-  Revenue sum of (Laptop, NY) over (Time)

1D Cuboids

-  Revenue sum of (Feb) over (products, locations)
-  Revenue sum of (CA) over (products, time)
-  Revenue sum of (Laptop) over (time, locations)

0D Cuboid

-  Revenue sum over (products, time, locations)

Multi-Dimensional DB 구축: R의 활용

▶ 다차원 Multi-dimensional DB(cube)의 생성

```
revenue_cube <- tapply(sales_fact$amount,  
                        sales_fact[, c("prod", "month", "year", "loc")],  
                        FUN=function(x){return(sum(x))})
```

```
> revenue_cube  
, , year = 2012, loc = CA  
  
      month  
prod    1     2     3     4     5     6     7     8     9    10    11    12  
Laptop  225   675  1125   NA   450   450   900   675   225   900   NA   675  
Printer  NA  1140   570   570   NA    NA  1710   NA    NA  1140  1710   570  
Tablet  4480  3360  7840  7840  4480  6720  4480  2240  7840  6720  4480  4480  
  
, , year = 2013, loc = CA  
  
      month  
prod    1     2     3     4     5     6     7     8     9    10    11    12  
Laptop  450   675   675   900   450   450   675   NA   450   675   900   NA  
Printer 1710   NA    NA   570   570   570  1140   NA    NA    NA  2280   570  
Tablet  4480  4480  5600  3360  4480  2240  4480  6720  2240  6720  1120  2240  
  
, , year = 2012, loc = NY  
  
      month  
prod    1     2     3     4     5     6     7     8     9    10    11    12  
Laptop  225   675   NA   450   225   900   225   675   900   225   225   NA  
Printer 1710   570   NA  1710   570  2850  1140  1140  1140   NA    NA  1140  
Tablet  1120  3360  2240  5600   NA  3360  2240  1120  2240  6720  8960  6720
```

○○○ tapply 연습

```
> sales_fact[1:20,]
  month year loc   prod unit amount
1     8  2012 ON Tablet    2   2240
2     7  2012 ON Tablet    1   1120
3     2  2012 NY Tablet    1   1120
4     3  2013 QU Tablet    1   1120
5     6  2013 WA Tablet    1   1120
6     3  2012 CA Tablet    2   2240
7     9  2012 CA Tablet    1   1120
8     6  2012 WA Laptop    1    225
9     7  2013 NY Tablet    1   1120
10    1  2013 QU Printer    2   1140
11    5  2012 ON Printer    1    570
12    4  2013 QU Laptop    1    225
13   11  2012 QU Tablet    1   1120
14    2  2013 WA Tablet    2   1120
15    6  2012 QU Printer    1   1120
16    1  2012 NY Tablet    1   1120
17    6  2012 NY Laptop    1   1120
18    5  2012 NY Printer    1   1120
19   10  2013 NY Tablet    1   1120
20    9  2013 NY Printer    1   1120
```

vector

첫 인자 vector와 동일한
길이의 index (factor)

```
> tapply(test$amount, test[, c("prod")], sum)
Laptop Printer Tablet
   900   3990  16800
```

SQL의 “group by” 연산과 유사

```
> tapply(test$amount, test[, c("prod", "loc")], sum)
      loc
prod   CA  NY  ON  QU  WA
Laptop  NA 450  NA 225 225
Printer  NA 1710 570 1710  NA
Tablet 3360 4480 3360 2240 3360
```

```
> tapply(test$amount, test[, c("prod", "loc", "year")], sum)
, , year = 2012
```

```
      loc
prod   CA  NY  ON  QU  WA
Laptop  NA 450  NA  NA 225
Printer  NA 570 570 570  NA
Tablet 3360 2240 3360 1120  NA
```

```
, , year = 2013
```

```
      loc
prod   CA  NY  ON  QU  WA
Laptop  NA  NA  NA 225  NA
Printer  NA 1140  NA 1140  NA
Tablet  NA 2240  NA 1120 3360
```

Multi-Dimensional DB 구축: R의 활용

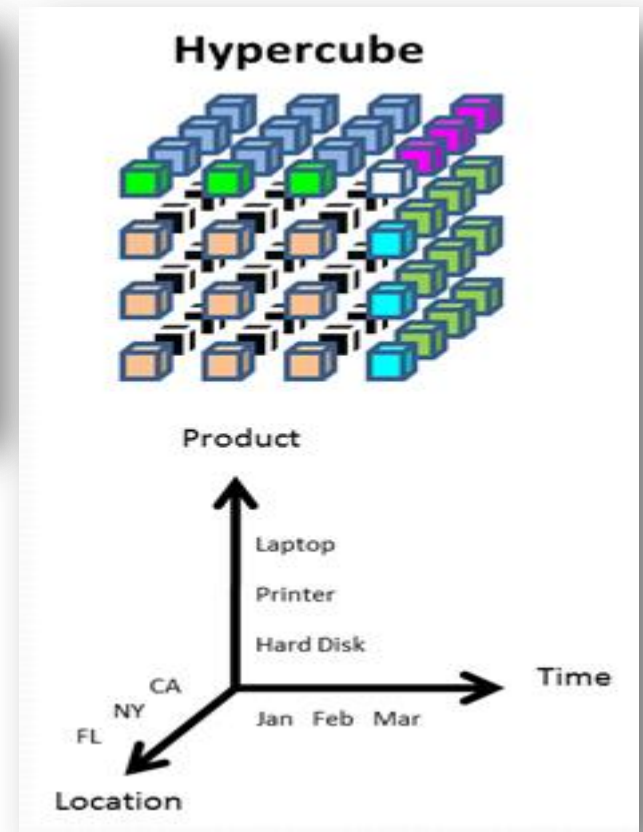
▶ `dimnames(revenue_cube)`

```
> dimnames(revenue_cube)
$prod
[1] "Laptop" "Printer" "Tablet"

$month
[1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12"

$year
[1] "2012" "2013"

$loc
[1] "CA" "NY" "ON" "QU" "WA"
```





Multi-Dimensional DB Operations



▶ **OLAP operations**

- ▶ Slice
- ▶ Dice
- ▶ Rollup
- ▶ Drilldown
- ▶ Pivot





Multi-Dimensional DB Operations



▶ Slicing

- ▶ # cube data in Jan, 2012
- ▶ revenue_cube[, "1", "2012",]

cube 차원 순서 : "prod", "month", "year", "loc "

- ▶ # cube data in Jan, 2012
- ▶ revenue_cube["Tablet", "1", "2012",]

```
> revenue_cube[, "1", "2012",]  
      loc  
prod   CA    NY    ON    QU    WA  
Laptop 225   225   225   225   NA  
Printer NA  1710   NA    NA    NA  
Tablet 4480 1120 1120   NA  1120
```

```
> revenue_cube["Tablet", "1", "2012",]  
      CA    NY    ON    QU    WA  
4480 1120 1120   NA  1120
```




Multi-Dimensional DB Operations



▶ Dicing

▶ `scube <- revenue_cube[c("Tablet","Laptop"),
c("1","2","3"), ,
c("CA","NY")] # dicing`



cube 차원 순서 : "prod", "month", "year", "loc "

▶ `scube[, "1", , "CA"]` # 작은 size의 cube에서 질의

```
> scube[ , "1", , "CA"]
      year
prod   2012 2013
Tablet 4480 4480
Laptop  225  450
```



Multi-Dimensional DB Operations



cube 차원 순서 : "prod", "month", "year", "loc "

▶ Roll-up

- ▶ 관찰하고 싶지 않은 차원을 없애기 위해 집계함수(aggregation function) sum 을 이용
- ▶ 예: year, prod, loc 차원에 대해 데이터 관찰 -> loc 차원 없애고, year, prod 차원에 대해 데이터 관찰

```
apply(revenue_cube, c("year", "prod"),  
      FUN=function(x){return(sum(x, na.rm=TRUE))})
```

(n차원) array

FUN함수를 적용하게 되는
기준 컬럼

	prod		
year	Laptop	Printer	Tablet
2012	23175	27360	192640
2013	20025	26790	170240



Multi-Dimensional DB Operations



▶ Drill-down

- ▶ 차원의 추가 또는 차원 하향 레벨링을 통해 세밀하게 데이터 관찰
- ▶ 예: 각 product에 대한 매년 판매량 관찰 -> 매달 판매량 관찰

```
apply(revenue_cube, c("year", "month", "prod"),  
      FUN=function(x){return(sum(x, na.rm=TRUE))})
```

```
, , prod = Laptop  
      month  
year   1    2    3    4    5    6    7    8    9   10   11   12  
 2012  900 4050 3150  900 1800 2475 2700 2025 1575 1575  450 1575  
 2013 1350  900 2250 1125 2250 2025 2475 1575 1800 1575 2025  675  
  
, , prod = Printer  
      month  
year   1    2    3    4    5    6    7    8    9   10   11   12  
 2012 1710 3990 1140 2850  570 4560 3990 1140 1140 2280 2280 1710  
 2013 2280 1710 3420 1140 2280 1710 2850 2280 1140 1710 5700  570  
  
, , prod = Tablet  
      month  
year   1    2    3    4    5    6    7    8    9   10   11   12  
 2012  7840 17920 15680 16800  6720 11200 17920 11200 19040 23520 26880 17920  
 2013 14560 20160 13440 14560 12320  6720 19040 17920  6720 23520 14560  6720
```

Multi-Dimensional DB Operations

▶ Pivoting

- ▶ 예: year vs. month (또는 month vs. year) 차원에 대한 판매량 관찰

```
apply(revenue_cube, c("year", "month"),  
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

	month											
year	1	2	3	4	5	6	7	8	9	10	11	12
2012	10450	25960	19970	20550	9090	18235	24610	14365	21755	27375	29610	21205
2013	18190	22770	19110	16825	16850	10455	24365	21775	9660	26805	22285	7965

- ▶ 예: product vs. location (또는 location vs. product) 차원에 대한 판매량 관찰

```
apply(revenue_cube, c("prod", "loc"),  
      FUN=function(x) {return(sum(x, na.rm=TRUE))})
```

	loc				
prod	CA	NY	ON	QU	WA
Laptop	12600	11025	6300	8550	4725
Printer	14820	18810	5130	8550	6840
Tablet	113120	97440	51520	43680	57120