
COMP9318 Tutorial 1

Wei WANG

The University of New South Wales

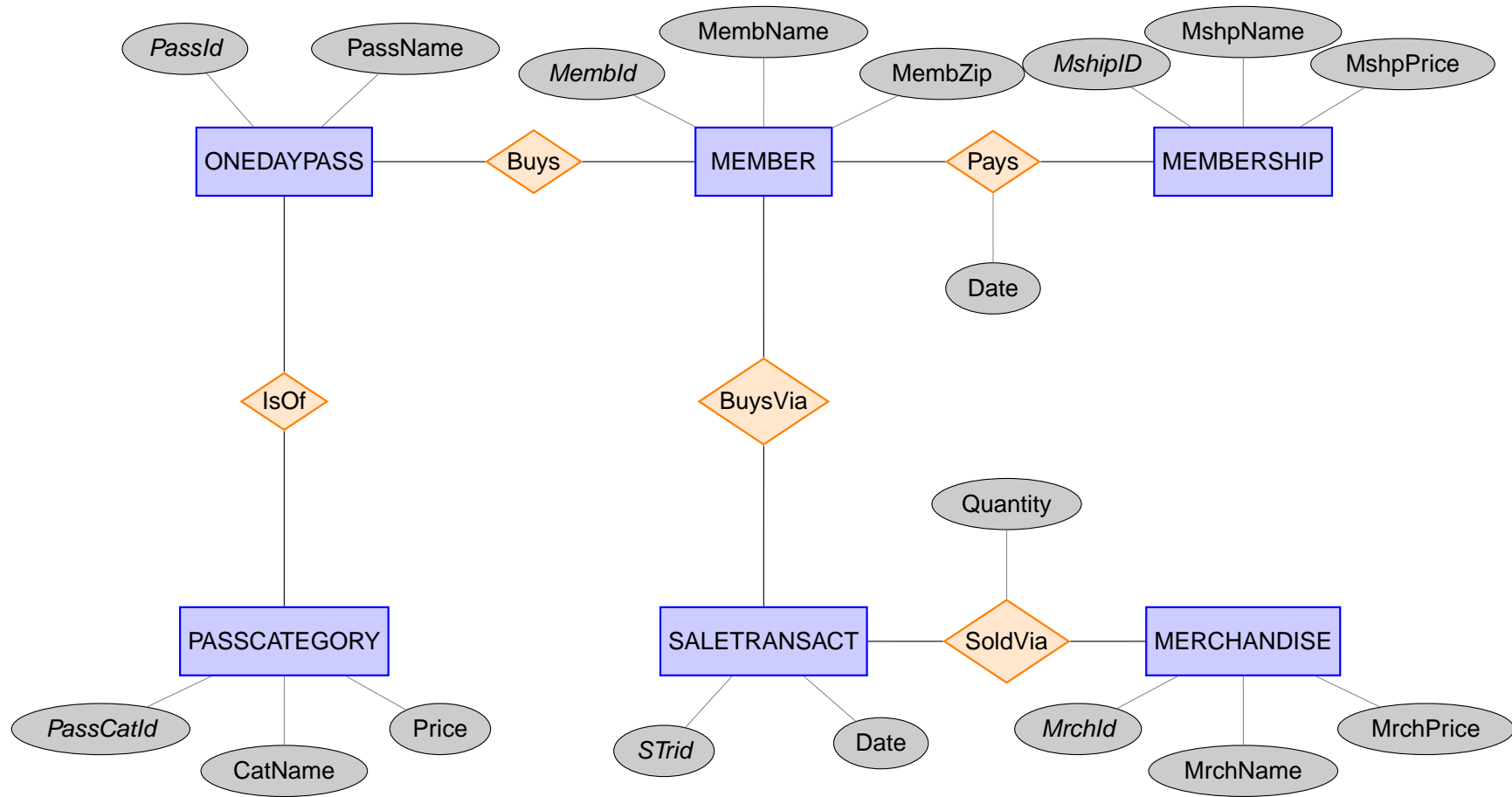
weiw@cse.unsw.edu.au

-
- ① Data Warehouse and OLAP
 - ② Data Preprocessing

Q1

- ① Create a star schema diagram that will enable FIT-WORLD GYM INC. to analyze their revenue.
 - ➔ The fact table will include — for every instance of revenue taken — attribute(s) useful for analyzing revenue.
 - ➔ The star schema will include all dimensions that can be useful for analyzing revenue
 - ➔ The only two data sources are shown below
- ② Appreciate the ETL process involved populating the data warehouse.
- ③ Appreciate the difference of formulating queries: “Find the percentage of revenue generated by members in the last year”.
- ④ How many cuboids are there in the complete data cube?

ER DIAGRAM



DATA INSTANCES

MEMBER

<i>Membid</i>	<i>MembName</i>	<i>MembZip</i>	<i>MshpID</i>	<i>MsDatePayed</i>
111	Joe	60611	M1	1-Jan-04
222	Mary	60640	M3	1-Jan-04
333	Sue	60611	M3	1-Jan-04

ONEDAYPASS

<i>PassID</i>	<i>PassDate</i>	<i>PassCatID</i>	<i>Membid</i>
1-001	1-Jan-04	PSA	111
1-002	1-Jan-04	PSA	333
1-003	2-Jan-04	PSK	333

PASSCATEGORY

<i>PassCatId</i>	<i>CatName</i>	<i>Price</i>
PSA	Adult	\$20
PSS	Senior	\$10
PSK	Kid	\$3

Note: MEMEBERS can bring in non-member guests. For each non-member guest, a member buys a one-day-guest-pass of a certain pass category.

MEMBERSHIP

<i>MshpID</i>	<i>MshpName</i>	<i>MshpPrice</i>
M1	Platinum	\$1,000
M2	Gold	\$800
M3	Value	\$300

MERCHANDISE

<i>MrchID</i>	<i>MrchName</i>	<i>MrchPrice</i>
AP1	T-shirt	\$11
AP2	Hat	\$9
EQ1	Jump Rope	\$12

SOLDVIA

<i>STrid</i>	<i>MrchID</i>	<i>Quatity</i>
11111	AP1	1
11112	AP2	1
11112	AP2	1
11113	EQ1	3

SALESTRANSACT

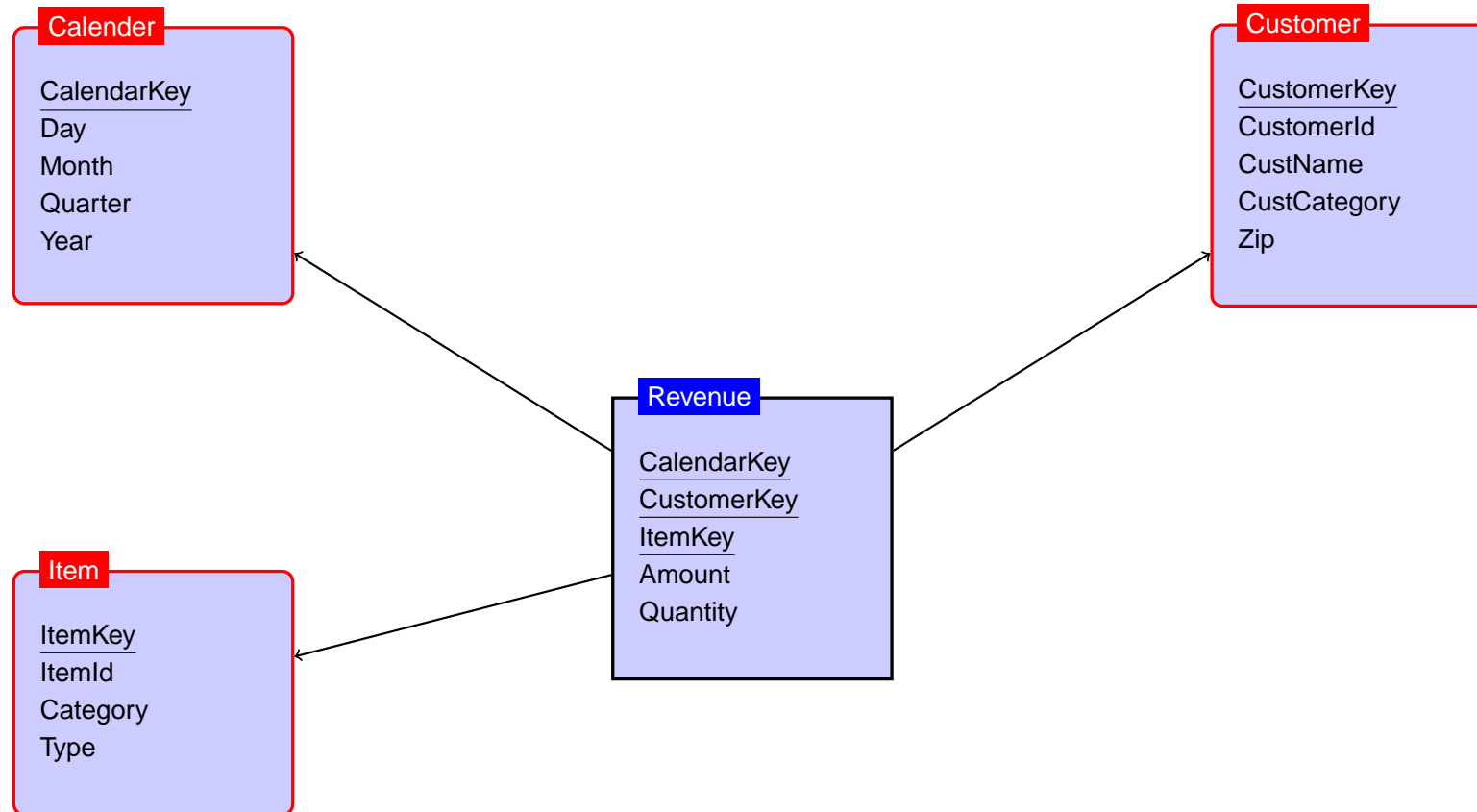
<i>STrid</i>	<i>Date</i>	<i>Membid</i>
11111	1-Jan-04	333
11112	2-Jan-04	222
11113	3-Jan-04	111

ANOTHER DATA SOURCE

SPECIALEVENT

<i>CorpCustID</i>	<i>CorpCustNameLoc</i>	<i>EventTypeCode</i>	<i>EvetType</i>	<i>EventDate</i>	<i>AmountCharged</i>
CC1	Sears, Chicago 60640	L-A	All Day Rental,	January 4, 2004	\$3500
CC2	Boeing, Chicago 60611	L-H	Half Day Rental,	January 5, 2004	\$2200

SOLUTION: STAR SCHEMA



POPULATED TABLES

CALENDAR				
<i>CalendarKey</i>	<i>Day</i>	<i>Month</i>	<i>Quarter</i>	<i>Year</i>
1	1	Jan	1	2004
2	2	Jan	1	2004
3	3	Jan	1	2004
4	4	Jan	1	2004
5	5	Jan	1	2004

REVENUE				
<i>CalendarKey</i>	<i>CustKey</i>	<i>ItemKey</i>	<i>Amount</i>	<i>Quantity</i>
1	1	1	\$1000	1
1	2	3	\$300	1
1	3	3	\$300	1
1	1	4	\$20	1
1	3	4	\$20	1
2	3	6	\$3	1
1	3	7	\$11	1
1	3	8	\$9	1
2	2	8	\$9	1
3	1	9	\$36	3
4	4	10	\$3500	1
5	5	11	\$2200	1

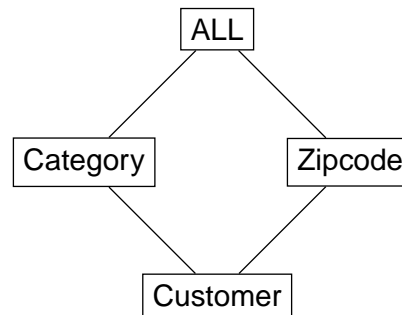
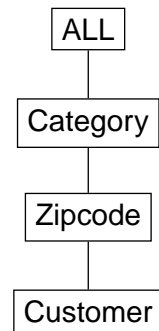
ITEM			
<i>ItemKey</i>	<i>ItemId</i>	<i>Category</i>	<i>Type</i>
1	M1	Memship	Platinum
2	M2	Memship	Gold
3	M3	Memship	Value
4	PSA	OneDayP.	Adult
5	PSS	OneDayP.	Senior
6	PSK	OneDayP.	Kid
7	AP1	Mrch.	T-Shirt
8	AP2	Mrch.	Hat
9	EQ1	Mrch.	Jump Rope
10	L-A	Spec. Evnt	All Day
11	L-H	Spec. Evnt	Half Day

CUSTOMER				
<i>CustKey</i>	<i>CustId</i>	<i>CustName</i>	<i>CustCategory</i>	<i>Zip</i>
1	111	Joe	Ind	60611
2	222	Mary	Ind	60640
3	333	Sue	Ind	60611
4	CC1	Sears	Corp	60640
5	CC2	Boeing	Corp	60611

-
- ① See above. Note that this is not the unique answer.
 - ② There are several tasks involved when importing the data into the data warehouse. E.g., we need to *extract* zipcode information from *CorpCustNameLoc*; we need to perform aggregation ($price \cdot Quantity$) for tuples in the *merchandise* table; we might also need to deal with (near) duplicate object detection (e.g., the same “member” that appear in two data sources).
 - ③ “Find the percentage of revenue generated by members in the last year” can be easily answered on the star schema by two aggregate queries on the fact table. Specifically, if the complete data cube has been built, the queries can be efficiently answered by the cuboid (*Year*), and the cuboid (*Year, Category*).
 - ④ Since *CustName* is not likely to be a good “level” for analysis (rather, it is a descriptive attribute), there are 4 levels on *Calendar* dimension, 3 on *Item*, and 3 on *Customer*. Therefore, there are $(4 + 1) * (3 + 1) * (3 + 1) = 80$ in total.

Note that we could have different hierarchies on a dimension. E.g., we

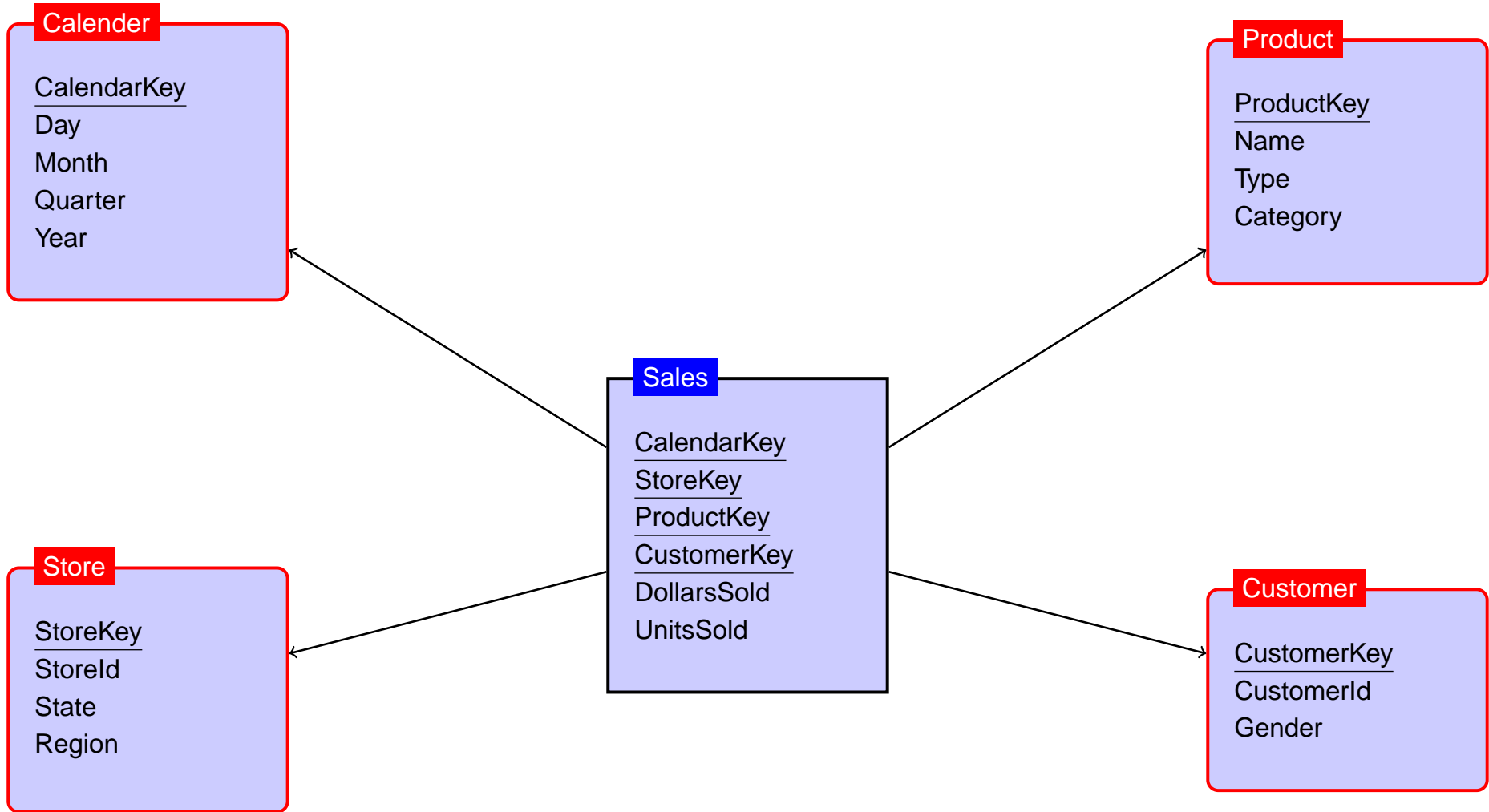
could consider the hierarchy on the Customer dimension one of the following. They have different semantics, but do not affect the number of cuboids.



Q2

Consider the star schema below

- Write an MDX query that display total DollarsSold for each product category and each store in the State 'CA'.
- create a star schema that has **Month** and **Region** as the finest granularity on the corresponding dimensions. Show all tables in the new data model populated with the data based on the data from the original model.



POPULATED TABLES

CALENDAR

<i>CalendarKey</i>	<i>Day</i>	<i>Month</i>	<i>Quarter</i>	<i>Year</i>
1	1	Jan	1	2003
2	2	Jan	1	2003
3	1	Feb	1	2003

STORE

<i>StoreKey</i>	<i>StoreID</i>	<i>State</i>	<i>Region</i>
1	X1	Maine	East
2	X2	New Jersey	East
3	Y1	Ohio	Midwest

SALES

<i>CalendarKey</i>	<i>ProKey</i>	<i>StoreKey</i>	<i>CustKey</i>	<i>\$Sold</i>	<i>UnitsSold</i>
1	1	1	1	\$15	1
1	2	2	2	\$20	1
1	2	1	1	\$40	2
1	2	2	1	\$20	1
2	2	1	1	\$19	1
2	2	2	1	\$19	1
3	3	3	1	\$9	2
3	3	3	2	\$9	1
3	3	3	3	\$9	1

PRODUCT

<i>ProKey</i>	<i>ProName</i>	<i>ProType</i>	<i>Category</i>
1	Luvs 50	Diapers	Infant Care
2	Huggies 24	Diapers	Infant Care
3	High C	Vitamin	Dietary Supp

CUSTOMER

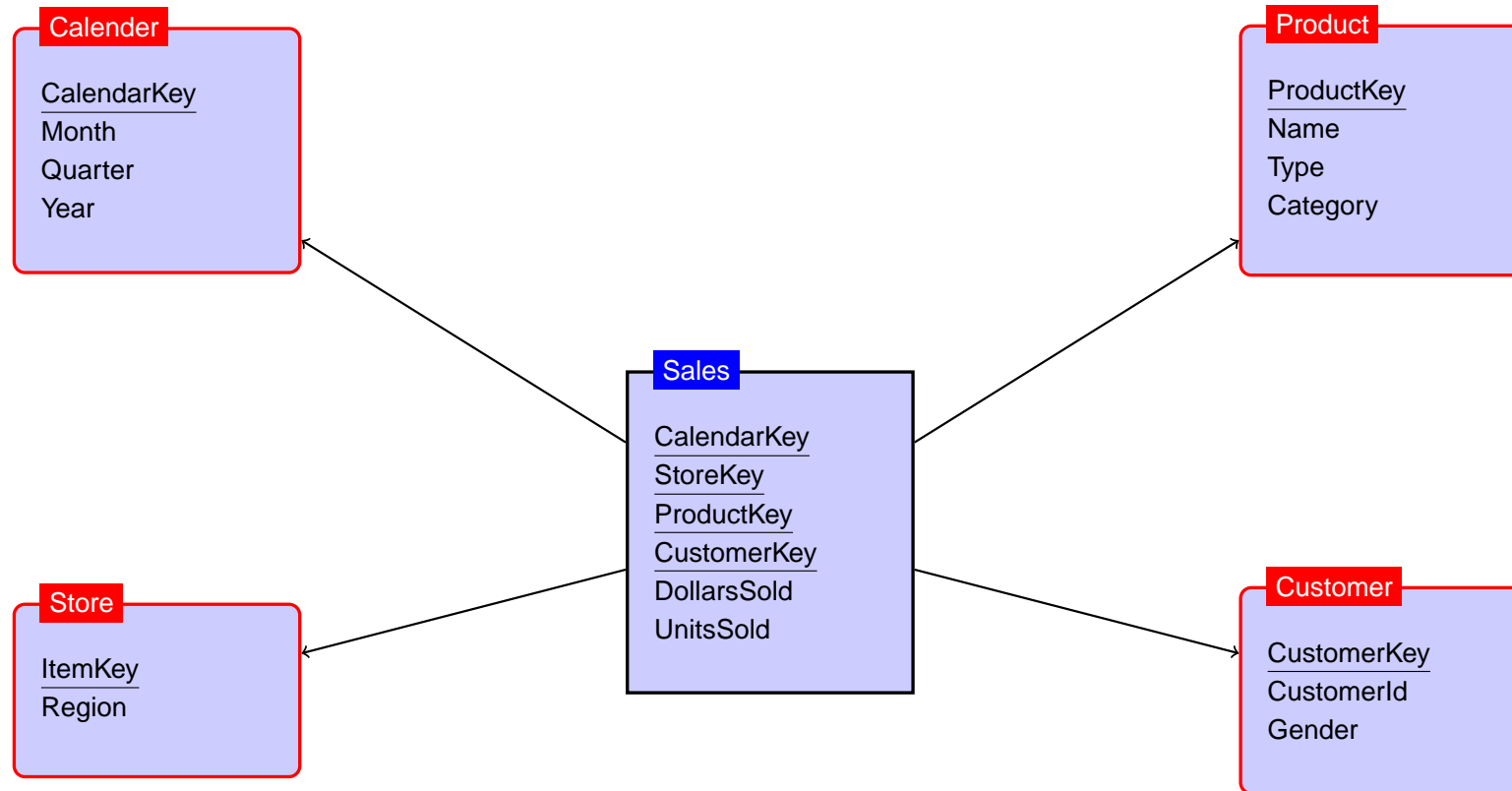
<i>CustKey</i>	<i>CustID</i>	<i>Gender</i>
1	12	Male
2	23	Male
3	34	Female

SOLUTION: MDX QUERY

```
SELECT [Product].[Category].MEMBERS ON COLUMNS
       [Store].[USA].[CA].CHILDREN ON ROWS
FROM   [Sales]
WHERE  ([Measures].[DollarsSold])
```

Q: what if we only want to query the sales data in 2007?

SOLUTION: STAR SCHEMA



POPULATED TABLES

CALENDAR

<i>CalendarKey</i>	<i>Month</i>	<i>Quarter</i>	<i>Year</i>
1	Jan	1	2003
2	Feb	1	2003

STORE

<i>StoreKey</i>	<i>Region</i>
1	East
2	Midwest

SALES (← Intermediate)

<i>CalendarKey</i>	<i>ProKey</i>	<i>StoreKey</i>	<i>CustKey</i>	<i>\$Sold</i>	<i>UnitsSold</i>
1	1	1	1	\$15	1
1	2	1	2	\$20	1
1	2	1	1	\$40	2
1	2	1	1	\$20	1
1	2	1	1	\$19	1
1	2	1	1	\$19	1
2	3	2	1	\$9	2
2	3	2	2	\$9	1
2	3	2	3	\$9	1

PRODUCT

<i>ProKey</i>	<i>ProName</i>	<i>ProType</i>	<i>Category</i>
1	Luvs 50	Diapers	Infant Care
2	Huggies 24	Diapers	Infant Care
3	High C	Vitamin	Dietary Supp

CUSTOMER

<i>CustKey</i>	<i>CustID</i>	<i>Gender</i>
1	12	Male
2	23	Male
3	34	Female

POPULATED TABLES

SALES

<i>CalendarKey</i>	<i>ProKey</i>	<i>StoreKey</i>	<i>CustKey</i>	<i>\$Sold</i>	<i>UnitsSold</i>
1	1	1	1	\$15	1
1	2	1	2	\$20	1
1	2	1	1	\$98	5
2	3	2	1	\$9	2
2	3	2	2	\$9	1
2	3	2	3	\$9	1

Q3

Suppose that the data for analysis include the attribute A having the following values: 1, 7, 7, 19, 13. The order among the data is important and thus you cannot sort them. Number of bins is 3. Ties can be broken arbitrarily.

1. Consider the following table.

# of items in a bin	1	7	7	19	13
1	0	0	0	0	0
2	18	?	?	?	n/a
3	?	?	?	n/a	n/a

Each entry in the table records the SSE (*Sum Square Error*) if a certain number of items are stored in a bin. More formally, for an entry in the i -th column in the j -th row (both i and j start from 1), it records the SSEs if the consecutive j values starting

at column i is put into one bin. For example, every entry in the first row is zero because there is no error (i.e., $SSE = 0$) if only one value is put into a bin. The first column in the second row is 18 because this is the SSE of a bin with values 1 and 7.

Fill in the rest of the entries in the table marked with “?”.

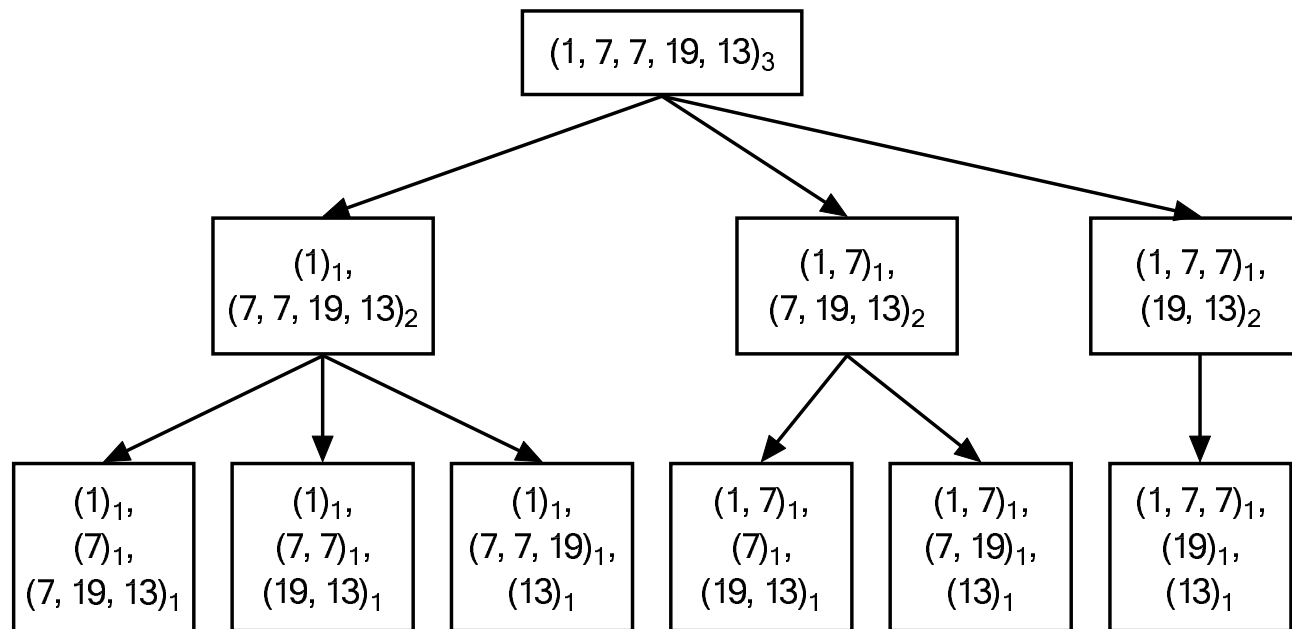
2. Construct the V-optimal histogram (which minimizes the SSE). You need to illustrate your steps.
3. MaxDiff is a heuristic algorithm to obtain a near-optimal histogram. Show an example such that it is possible that MaxDiff algorithm will give a histogram that is **not** optimal.

SOLUTION TO Q3

1. See the following table.

# of items in a bin	1	7	7	19	13
1	0	0	0	0	0
2	18	0	72	18	n/a
3	24	96	72	n/a	n/a

2. See the following figure.



Name each of the possible solutions in the leaf of the figure as (a) to (f). By looking up the table above, it is easy to identify that the costs are:

solution	cost
(a)	$0 + 0 + 72 = 72$
(b)	$0 + 0 + 18 = 18$
(c)	$0 + 96 + 0 = 96$
(d)	$18 + 0 + 18 = 36$
(e)	$18 + 72 + 0 = 90$
(f)	$24 + 0 + 0 = 24$

3. we apply MaxDiff on the given dataset. The gaps in the data are: 6, 0, 12, 6. For 3 bins, we need to choose 2 gaps. Since there is a tie, we may break it arbitrarily. If we take the first 6, this will give us (1), (7, 7), (19, 13), with SSE = 18. If we take the second 6, this will give up (1, 7, 7), (19), (13), with SSE = 24. Therefore, it is possible that MaxDiff may give a non-optimal solution.

Q4

Calculate the edit distance between the following pairs of strings:

1. "abcd" and "cdab"
2. "abcdef" and "cdabfe"

SOLUTION TO Q4

(omitted)

Q5

Consider performing the prefixed-based similarity join on the following tables R using an edit distance threshold of 1 and 2-grams (aka. bi-grams) without considering the beginning/end of the string characters (i.e., $\sim/\$$).

R	
1	report
2	repert
3	raport
4	reporter

SOLUTION TO Q5

① Bi-grams for R are listed below

R	
1	{ re, ep, po, or, rt }
2	{ re, ep, pe, er, rt }
3	{ ra, ap, po, or, rt }
4	{ re, ep, po, or, rt, te, er }

② Next, we need to determine a global order \mathcal{O} according to decreasing IDF order (i.e., from rarest to the most frequent). We calculate DF as followings

bi-gram	re	ep	po	or	rt	pe	er	ra	ap	te
DF	3	3	3	3	4	1	2	1	1	1

Then we sort the bi-grams according to the increasing DF order (break the tie consistently, e.g., using the dictionary order of the

bi-grams)

Order: ap, pe, ra, te, er, ep, or, po, re rt.

R	
1	{ ep, po, or, re, rt }
2	{ pe, er, ep, re, rt }
3	{ ap, ra, or, po, rt }
4	{ te, er, ep, po, or, re, rt }

③ Compute the length-3 prefixes for each record.

$Prefix(R)$	
1	{ ep, po, or }
2	{ pe, er, ep }
3	{ ap, ra, or }
3	{ te, er, ep }

④ We start from the 1st record. Its candidate set is empty and we only

need to index its own prefix.

The inverted index is:

<u>$INV(Prefix(R))$</u>	
ep	1
po	1
or	1

- ⑤ We move to the 2nd record. “ep” gives a candidate 1. We need to verify the candidate by computing the edit distance between records 1 and 2, which turns out to be 1. We then index 2nd record’s prefix.

The inverted index is:

$INV(Prefix(R))$

ep 1, 2

po 1

or 1

pe 2

er 2

- ⑥ We move to the 3rd record. “or” gives a candidate 1. We need to verify the candidate by computing the edit distance between records 1 and 3, which turns out to be 1. We then index 3rd record’s prefix.

The inverted index is:

$INV(Prefix(R))$

ep 1, 2

po 1

or 1, 3

pe 2

er 2

ap 3

ra 3

- ⑦ We move to the 4th record. “ep” gives candidates 1 and 2. We need to verify the candidate by computing the edit distance between records (1, 4) and (2, 4), respectively; both are larger than 1.
- ⑧ So the final results are: (1, 2), (1, 3).

Note that if we also use length filtering, the candidate set for the 4th record will be empty, as none of the preceding record has a

length within $[8-1, 8+1]$.