Unit 11
Online Analytical Processing (OLAP)
Basic Concepts

OLAP in Context

**User Level
(View Level)**

Derived Tables
Constraints, Privileges ——— Queries (DML)

↑ Derived

**Community Level
(Base Level)**

Base Tables
Constraints, Privileges

— Application Data Analysis (ER)
— Normalization (NFs)
— Schema Specification (DDL)
— Queries (DML)

⇓ Implemented

Physical Level

Files
Indexes, Distribution ——— Query Execution (B^+, …, Execution Plan)

⇓ Relies on

DBMS OS Level

Concurrency

Recovery ——— Transaction Processing (ACID, Sharding)

⇓ Runs on

**Centralized
Or
Distributed**

Standard OS
Standard Hardware

OLAP vs. OLTP

◆ We have focused until now on **OLTP**: Online Transaction Processing

◆ This dealt with storing data both logically and physically and managing transactions querying and modifying the data

◆ We will now focus providing support for analytical queries, essentially statistical and summary information for decision-makers, that is on **OLAP**: Online Analytical Processing

◆ This may be accomplished by preprocessing, for efficiency purposes, and producing special types of views, which are also not necessarily up to date

  • Not up to date may not be a problem in OLAP

◆ Data for OLAP (and more generally for data mining) is frequently stored in a **Data Warehouse**

Example

◆ Our company has several stores and sells several products

◆ The stores are in different locations

◆ The locations, identified by (city,state) pairs are grouped into several regions

◆ We divide the time of sale into four quarters

◆ The quarters are grouped into two half-years

Our Company

Store	Store#	City	State	Region
	Alpha	New York	NY	NE
	Beta	Albany	NY	NE

Quarter	Quarter#	Half_Year
	1	First
	2	First
	3	Second
	4	Second

Product	Product#	Weight	Price
	Book	4	100
	Glass	15	200

Our Sales

| Sale	Store#	Product#	Quarter#	$
	Alpha	Book	1	70,000
	Alpha	Glass	1	90,000
	Beta	Book	1	90,000
	Beta	Glass	1	80,000
	Alpha	Book	2	90,000
	Alpha	Glass	2	90,000
	Beta	Book	2	60,000
	Beta	Glass	2	50,000
	Alpha	Book	3	60,000
	Alpha	Glass	3	80,000
	Beta	Book	3	50,000
	Beta	Glass	3	90,000
	Alpha	Book	4	50,000
	Alpha	Glass	4	50,000
	Beta	Book	4	70,000
	Beta	Glass	4	70,000

Star Schema

◆ We want to support queries useful for statistical analysis by computing various sums, averages, etc.

◆ The structure we have is a ***star schema***

◆ In the middle we have our ***facts table***

Store	
PK	Store#
	City State Region

Sale	
PK,FK1 PK,FK2 PK,FK3	Quarter# Product# Store#
	$

Product	
PK	Product#
	Weight Price

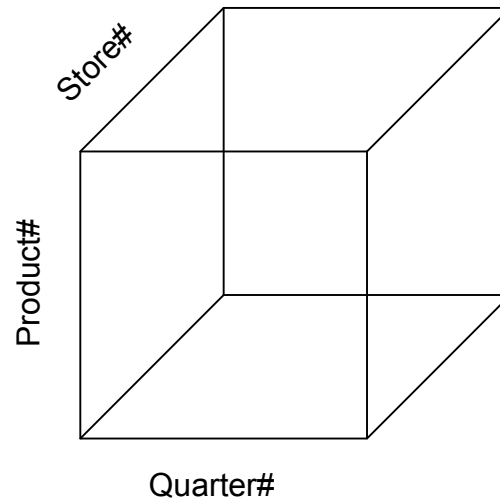| Quarter	
PK	Quarter#
	Half_Year
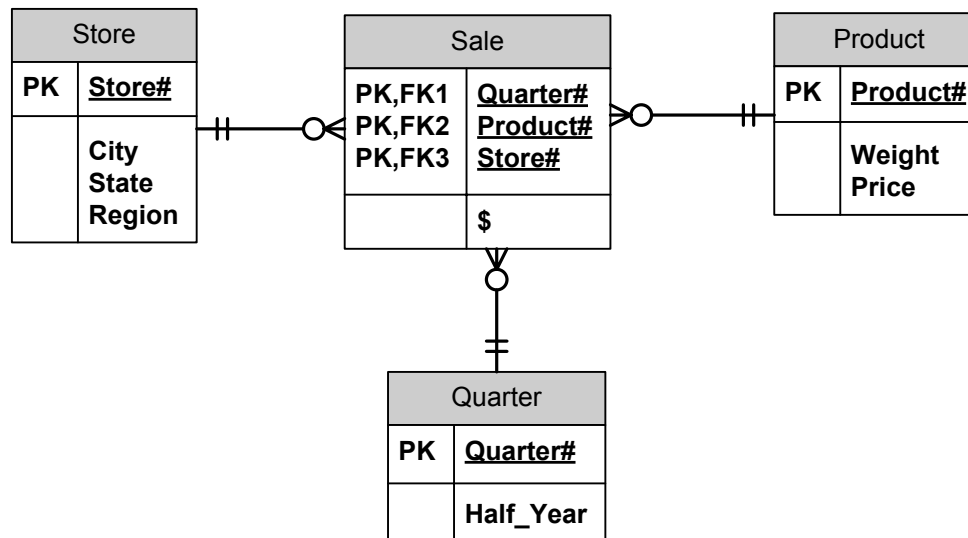
Snowflake Schema: Normalized Star Schema

◆ One could also normalize, as table Store is not normalized, since State → Region

◆ Then, one could get, which we will not consider further, a **snowflake schema**
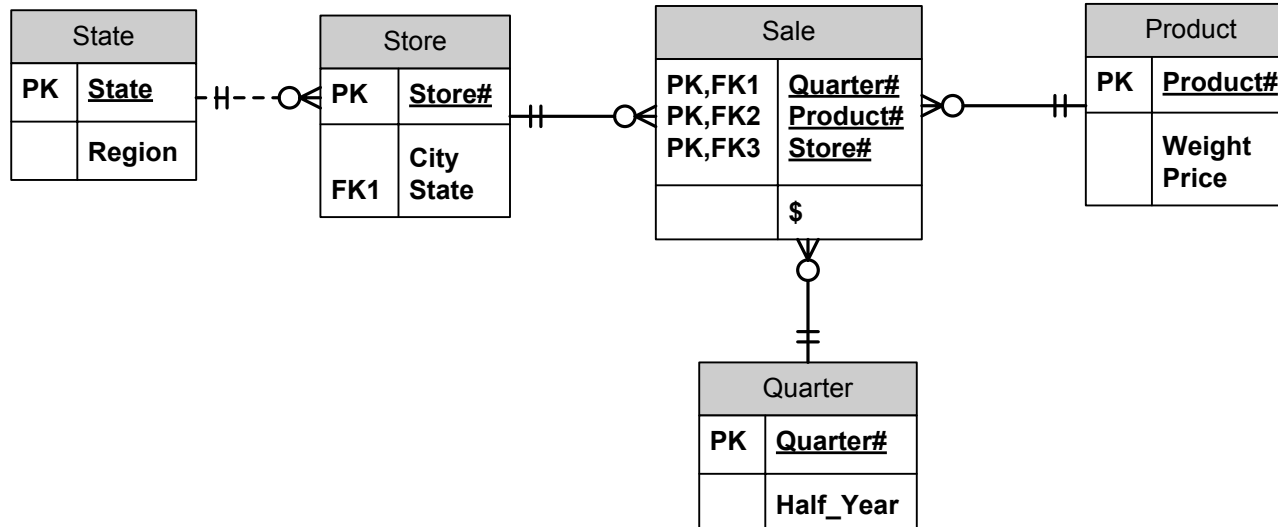
Cube

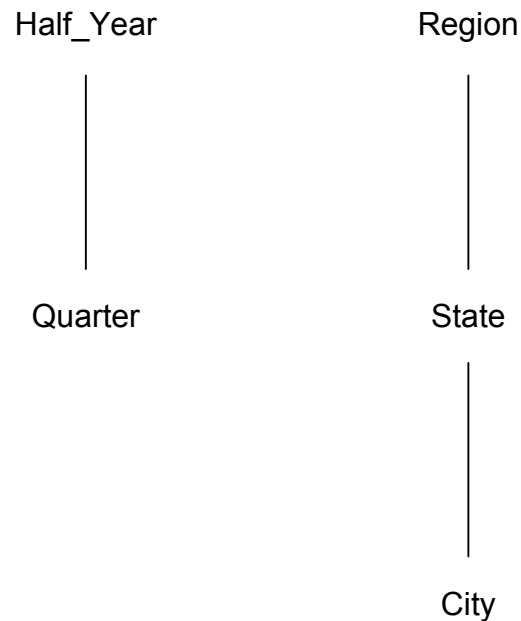◆ We could think of each row of fact table as occupying a voxel (volume element) in a ***cube***

Store#

Product#

Quarter#

◆ Cube, in general, can have any number of dimensions; in our example there are three

◆ This cube can then be ***sliced and diced***

Slice

◆ SELECT Store#, Product#, SUM($)
FROM Sale
GROUP BY Store#, Product#

◆ We can do all kinds of such slices

Dimension Hierarchies

◆ Dimensions could have hierarchies (or more generally even lattices)

◆ We have two very simple hierarchies

- One temporal: quarters are in half years
- One geographical: cities are in states are in regions

```
Half_Year          Region

   |                  |
   |                  |
   |                  |

Quarter            State

                      |
                      |
                      |

                    City
```

Using Hierarchies

◆ SELECT Sale.Product#, Quarter.Half_Year, SUM($)
FROM Sale, Quarter
WHERE Sale.Quarter# = Quarter.Quarter#
GROUP BY Half_Year;

◆ Will produce summaries by half years, not quarters

New Operator: CUBE

◆ SELECT Store#, Product#, SUM($)
FROM Sale
GROUP BY CUBE (Store#,Product#);

◆ Will produce all possible aggregations based on subsets of {Store#,Product#}, best explained by looking at what will come out

	Store#	Product#	$
	Alpha	Book	270,000
	Alpha	Glass	310,000
	Beta	Book	270,000
	Beta	Glass	290,000
	Alpha	NULL	580,000
	Beta	NULL	560,000
	NULL	Book	540,000
	NULL	Glass	600,000
	NULL	NULL	1,140,000

New Operator: ROLLUP

◆ ROLLUP produces only some of the aggregate operators produced by CUBE, best explained by example

◆ SELECT Store#, Product#, SUM($)
FROM Sale
GROUP BY ROLLUP (Store#,Product#);

	Store#	Product#	$
	Alpha	Book	270,000
	Alpha	Glass	310,000
	Beta	Book	270,000
	Beta	Glass	290,000
	Alpha	NULL	580,000
	Beta	NULL	560,000
	NULL	NULL	1,140,000

ROLAP And MOLAP

◆ ROLAP: Relational OLAP

◆ That is what we have been doing: OLAP information was stored as a set of star (or more generally snowflakes) schemas

◆ MOLAP: Multidimensional OLAP

◆ Information not stored as a relational database, but essentially as a cube

Oracle

◆ Oracle supports OLAP

◆ Zhaohui Zhang ran the example on our Oracle installation

◆ You can see the various scripts and results in Extras for this unit which will be put up later in the week

◆ Meanwhile, here are the actual SQL code fragments

Oracle: Defining the Database

```
create table store(
   sid char(20) primary key,
   city char(20),
   state char(20),
   region char(20)
);

create table product(
   pid char(20) primary key,
   weight number,
   price number
);

create table quarter(
   qid number primary key,
   half_year char(10)
);
```

Oracle: Defining the Database

```sql
create table sale(
   sid char(20),
   pid char(20),
   qid number,
   profit number,
   primary key(qid, pid, sid),
   foreign key(qid) references quarter(qid),
   foreign key(pid) references product(pid),
   foreign key(sid) references store(sid)
);
```

Oracle: OLAP Query

```
select sid, pid, sum(profit)
from sale
group by rollup(sid, pid);

select sid, pid, sum(profit)
from sale
group by cube(sid, pid);
```

Key Ideas

- ◆ OLAP vs. OLTP
- ◆ Star schema
- ◆ Snowflake schema
- ◆ Cube
- ◆ Slicing and dicing
- ◆ Dimension hierarchies
- ◆ ROLAP
- ◆ MOLAP