Lecture 13

Back up files also need to be encrypted

- Remember that if you drop one table, you cannot restore it from a physical backup. It's all or nothing.
  - *So physical and logical backup need not be seen as exclusive.*

Log shipping

> Remember that the log files (journal files) are files that are synchronously written whenever a transaction commits. They are used for bringing back database files to the image of memory after a crash.

The idea is to periodically ship log files to a distant server that hosts a copy of the database and

reapply them there.

Metrocluster

SAN systems manufacturers may offer metroclusters which may be disk-box-to-disk-box, physical replication that does only care about bytes, and not applications. It can be synchronous or asynchronous.

> Basically, remember that if you want zero loss, replication must be synchronous

Arbitrage – *you can't have everything*

Human errors can happen .But if somebody does a big mistake on the primary database, the mistake may travel to the backup database even before you notice the problem.

If problems are transferred too fast, you get **two corrupted databases** instead of one, and must restore from backups.

> People hate to say that, but there is always a part of "bets" in solutions that are adopted. You can think of hybrid modes: transferring logs as soon as possible, but waiting a little before reapplying them. But how long should you wait? And it will impact the time to restart operations if you need to reapply logs in a hurry.

Decision Supporting System

OLTP(On Line Transaction Processing)

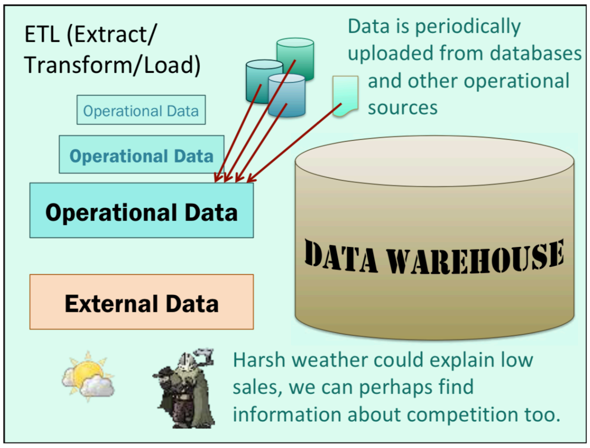
> Traditionally, systems that record orders, production, billing and so forth are called OLTP systems. They are characterized by short transactions that are executed a large number of times.

Don't maintain history records

Consider **Long Term** and **Additional Data**, we'll have to deal with possibly far more data than operational databases. We need to **aggregate**.

Data Warehouse

> This repository should be distinct from operational databases not only because the schema might be different, but because the query patterns will be very different, with almost no repeated queries (or repeated only once a day, week or month)



ETL (Extract/ Transform/Load)

Data is periodically uploaded from databases and other operational sources

Operational Data

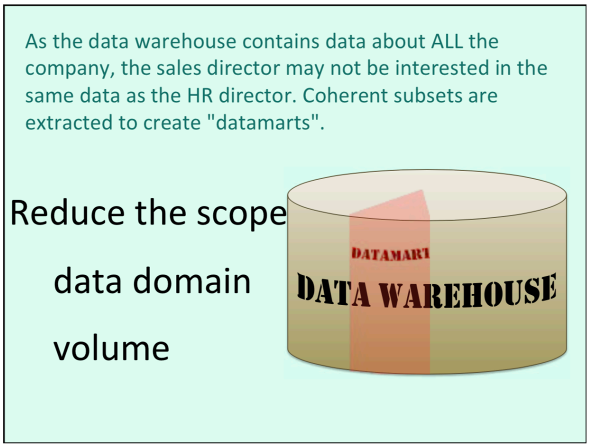
Operational Data

Operational Data

DATA WAREHOUSE

External Data

Harsh weather could explain low sales, we can perhaps find information about competition too.



As the data warehouse contains data about ALL the company, the sales director may not be interested in the same data as the HR director. Coherent subsets are extracted to create "datamarts".

Reduce the scope

data domain

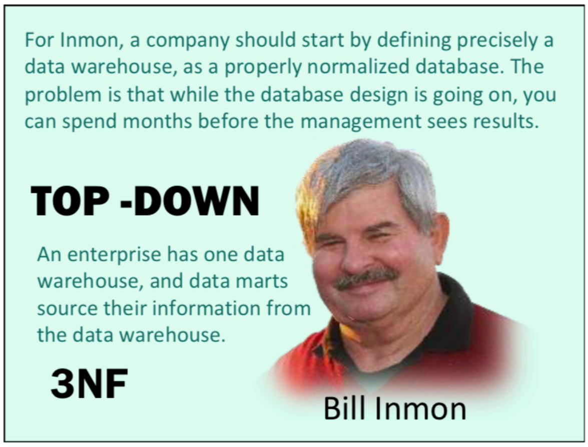
volume

DATAMART

DATA WAREHOUSE

not "real time" data uploading schedule

Break 3NF



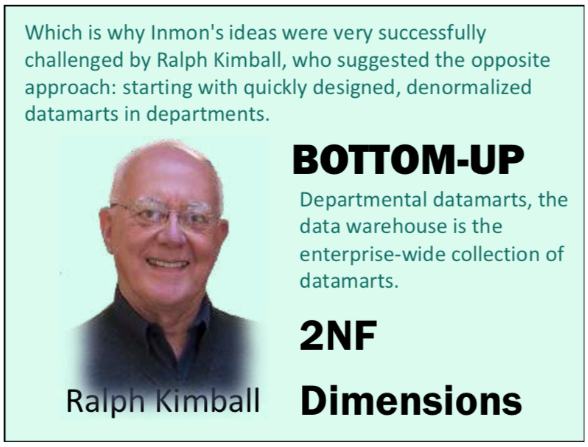
For Inmon, a company should start by defining precisely a data warehouse, as a properly normalized database. The problem is that while the database design is going on, you can spend months before the management sees results.

TOP -DOWN

An enterprise has one data warehouse, and data marts source their information from the data warehouse.

3NF

Bill Inmon



Which is why Inmon's ideas were very successfully challenged by Ralph Kimball, who suggested the opposite approach: starting with quickly designed, denormalized datamarts in departments.

BOTTOM-UP

Departmental datamarts, the data warehouse is the enterprise-wide collection of datamarts.

2NF

Dimensions

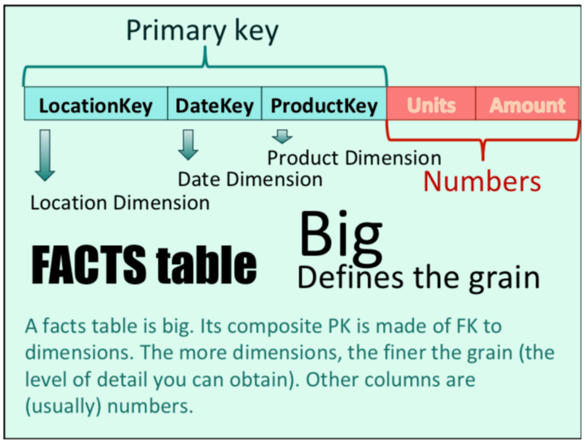
Ralph Kimball

Facts and Dimensions

I'd like to see sales in USD per country, month and per line of product.

This is a typical managerial request. Sales are facts. Everything that follows a "per" is a dimension. Simple.

Flickr: Steve Wilson



Primary key

LocationKey	DateKey	ProductKey	Units	Amount

Location Dimension

Date Dimension

Product Dimension

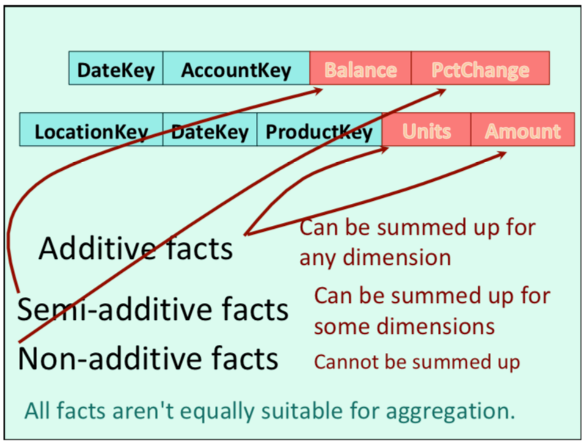
Numbers

FACTS table

Big

Defines the grain

A facts table is big. Its composite PK is made of FK to dimensions. The more dimensions, the finer the grain (the level of detail you can obtain). Other columns are (usually) numbers.



DateKey	AccountKey	Balance	PctChange

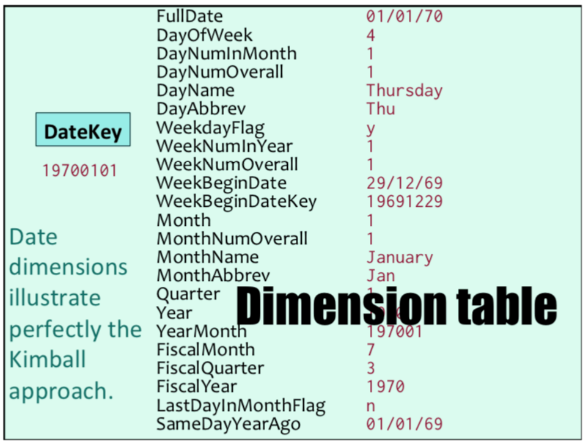
LocationKey	DateKey	ProductKey	Units	Amount

Additive facts — Can be summed up for any dimension

Semi-additive facts — Can be summed up for some dimensions

Non-additive facts — Cannot be summed up

All facts aren't equally suitable for aggregation.



DateKey
19700101

Date dimensions illustrate perfectly the Kimball approach.

FullDate	01/01/70
DayOfWeek	4
DayNumInMonth	1
DayNumOverall	1
DayName	Thursday
DayAbbrev	Thu
WeekdayFlag	y
WeekNumInYear	1
WeekNumOverall	1
WeekBeginDate	29/12/69
WeekBeginDateKey	19691229
Month	1
MonthNumOverall	1
MonthName	January
MonthAbbrev	Jan
Quarter	1
Year	[illegible]
YearMonth	197001
FiscalMonth	7
FiscalQuarter	3
FiscalYear	1970
LastDayInMonthFlag	n
SameDayYearAgo	01/01/69

Dimension table

> Kimball's reasoning is the following: if I store a date column, that will be difficult to query. If I want to aggregate by date or month, I'll need to apply (complicated and different in all DBMS products) date functions that I'll never be able to index because most date functions are NOT determininistic. The day is likely to be my smallest time unit. What are 20 years, with one row per day? Under 7,000 rows? Very tiny today. Let's have one row per date, and decline each date under every possible form, and index every column.