

CS307

Database Principles

Stéphane Faroult
faroult@sustc.edu.cn

朱悦铭 Zhu Yueming zhuym@sustc.edu.cn

REMINDER

Data in backup = data in production database

On the chapter of security, perhaps it is worth reminding that backups contain the same data as the database and that making backup files publicly readable is a bad idea. Some recent big data breaches were simply stealing backup files. You can also encrypt them.

PROTECT FILES

Remember that if you drop one table, you cannot restore it from a physical backup. It's all or nothing.

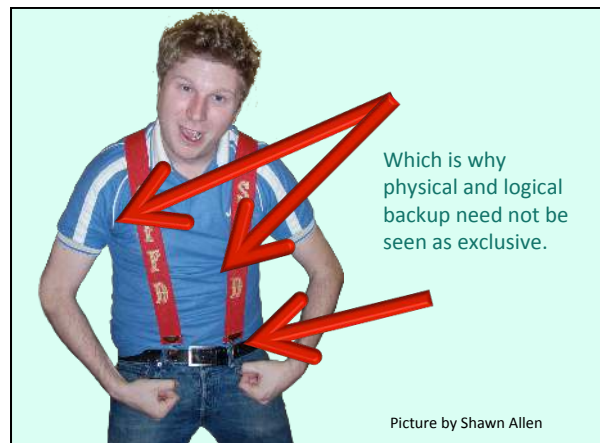
```
SQL> drop table super_important_data;
```

Table dropped.

```
SQL>
```

ooops !!!

If the database is big and the table relatively small, the full restore may be very disruptive.



RECYCLE BIN



For tables, Oracle implements a "recycle bin" that works like the trash can in Windows. You can "undrop" a table if this functionality is activated. Of course, as dropped tables are still physically there, you need to purge the "recycle bin" from time to time.

ORACLE

And (IMPORTANT!)

PRACTICE RECOVERY

Make sure you can do it at 2am with bleary eyes, knowing that the sun may not rise if you fail. Automate as much as you can.

So, we have a backup ...

Finding the right backup schedule is important. It boils down to some important questions: how fast do you need to have your database up and running after a major problem?

What about
AVAILABILITY?
or in other words ...

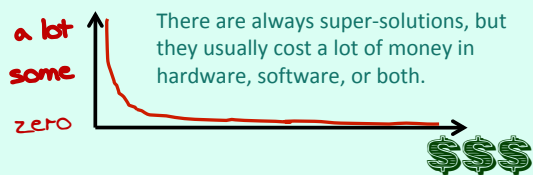
How **long** can you wait?



Flickr: Bev Goodwin

What **data loss** can you bear?

The other question is whether you need to get back absolutely everything or something consistent and recent enough.



In any system, you always want to eliminate "single points of failure", these critical elements that knock the whole information system out when they fail.

SINGLE POINT OF FAILURE

These is why you mirror disks: if a disk fails, you have another copy and can hot-plug a new disk to replace the failing one. No system interruption, no lengthy recovery of a past backup.



Every component must be checked, the impact measured if it fails, and if you can afford it it should be duplicated.

Cheaper disks?

CPU?

Memory?

Various cards (network, I/O controller)?

Power supply?

If anything fails you want to be able to switch, as transparently as possible, to a backup system that may possibly be slightly less performant but can ensure the main functionalities.

This is the concept of "High Availability" (often referred to as HA)

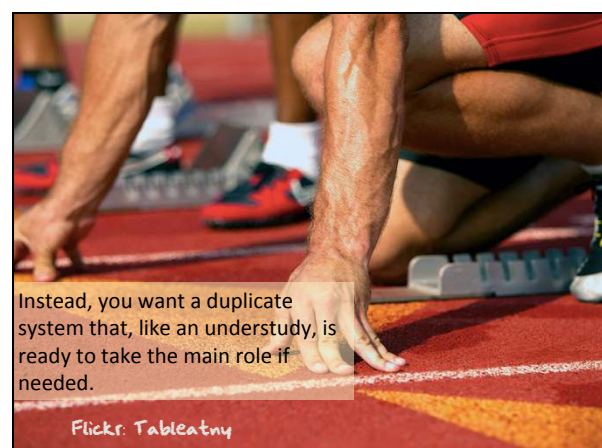
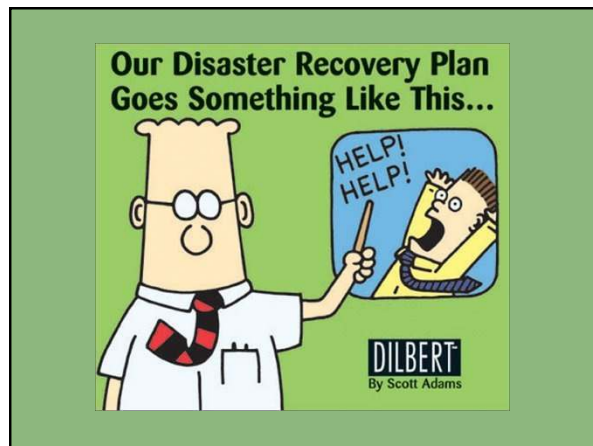


A typical HA solution is a cluster of computers that may or not access the same database files.

HA is only part of the story. HA is mostly about not having to say to your customers or colleagues "We are sorry, the system is out of order, we'll be back in a few hours".

Disaster Recovery Business Continuity

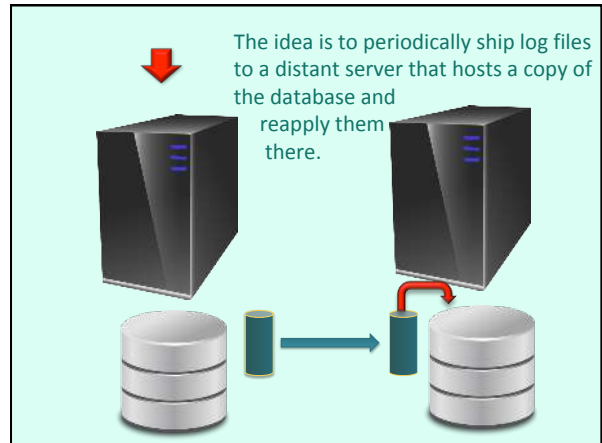
Disaster recovery is about fortunately far rarer events than hardware failures, but catastrophic events that can jeopardize the life of a company.



A commonly used method for keeping a server in the starting-blocks is 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.



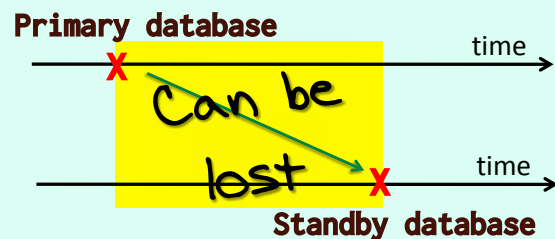
If the DBMS supports it, it can be relatively easy to script and various OS utilities can even help keeping two distant servers in sync.



ssh
rsync

Beware though that if all "ordinary" DML changes travel well from server to server, DDL operations are not always logged and may require special care.

With log shipping, you can reduce dramatically what can be lost to the content of logs that haven't yet been transferred.



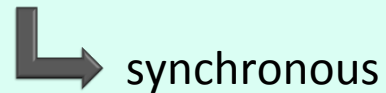
Metrocluster

Other solutions are available.

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.

In that case replication is transparent for the DBMS software.

zero data loss



synchronous

Performance issues

Basically, remember that if you want zero loss, replication must be synchronous - you receive the acknowledgement (in other words, the commit() call returns) only when local and distant disks have been written. OK if you replicate 10 km away, not so OK when it's 100 or 200 km away.

Hardware
or
Software
replication



And once again, you have the choice between hardware and software replication. Opinions vary about the virtues of the one or the other. Hardware replication is at a lower level and possibly more efficient, but what about replicating a corrupted block?



Finally, an attractive backup solution for smaller companies is backing up in "the cloud" – but there may be security issues if you don't encrypt your files.

Practical problem with disaster recovery

Disaster recovery, by definition, occurs in the middle of a disaster, which means complete panic. Database administrators and system managers can expect to have to switch systems in the middle of phone calls, and, if they don't answer, nervous managers around them asking when the system will be up. Try to imagine what it can be when computer systems crash at a busy airport. Recovery procedures must be as automated (and tested) as possible, because it's very hard to think straight in these cases.

Experience proves though that when the time comes of switching to a remote backup system, the most consuming may not be starting everything "for real" at the backup site and routing all applications there.

Switching a system is a big decision, and between the time when a problem is detected, its severity assessed

Switch and the "go" from management, quite a lot of time can fly away.

Diagnosis?

Decision taking?

One point often overseen is that the goal of a backup site isn't to replace the primary site forever. When the primary site is finally back on its feet, you must restore recent copies of databases, reset log shipping but the other way round, and finally, when ready, switch again (this time it can be scheduled at a low activity time), and reset the backup site to a backup role. All operations that aren't necessarily obvious.

Think about
return to normal ...

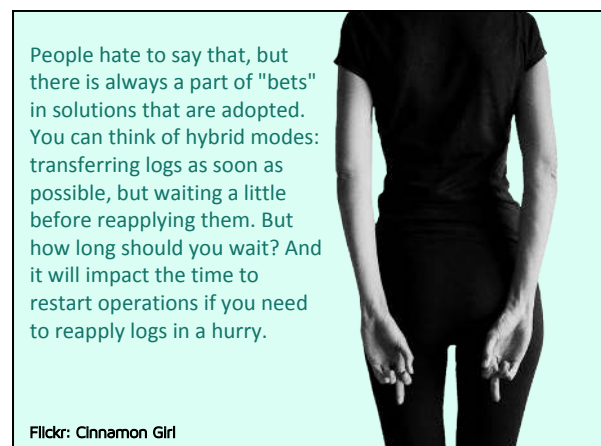
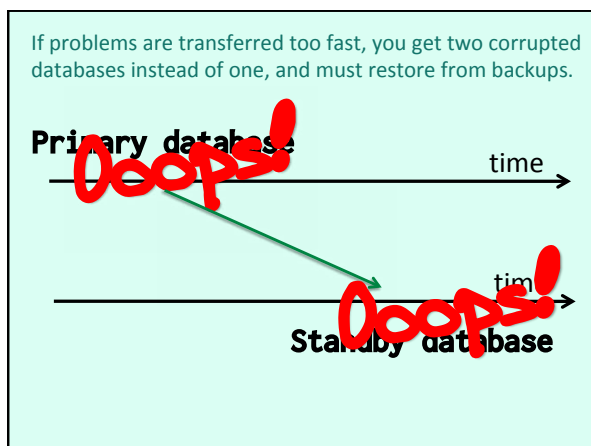
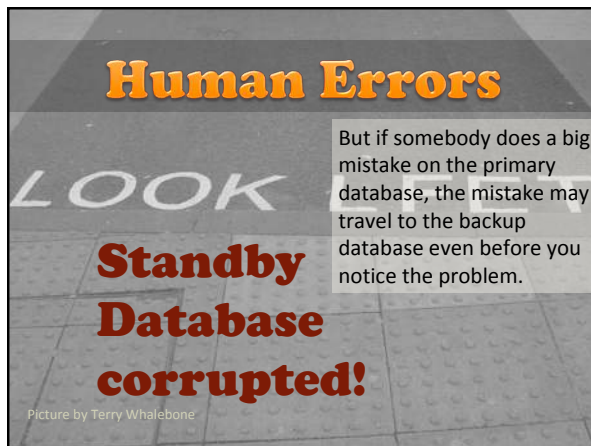
Hardware Failure



Try to
minimize lag

When you want protection against hardware failures, you want the backup site to run as close as possible behind the primary site.

Picture by Philip Bitner



Whatever happens, the worse possible day when to check that you can switch to another database or restore, and how fast you can do it, is the day when a serious problem happens. Every procedure should be carefully tested and documented.



To remember

Logical vs physical backup

Cold vs hot backup

Synchronous vs asynchronous

Replication

Arbitrage – you can't have everything

DECISION SUPPORT SYSTEMS

DataWarehousing, Business Intelligence and all that.

We have been talking mostly about operational systems; let's talk about another important aspect of information systems, decision support systems that are supposed to provide "strategical" and not only "tactical" data.



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.

Getting information out of OLTP databases isn't easy: you have tons of tables, queries are complicated, answering an unusual question is difficult.

“

During the turmoil many banks had to carry out big fact-finding missions to see where they stood. "Answering such questions as 'What is my exposure to this counterparty?' should take minutes. But it often took hours, if not days," says Peyman Mestchian, managing partner at Chartis Research, an advisory firm. Insiders at Lehman Brothers say its European arm lacked an integrated picture of its risk position in the days running up to its demise.

”

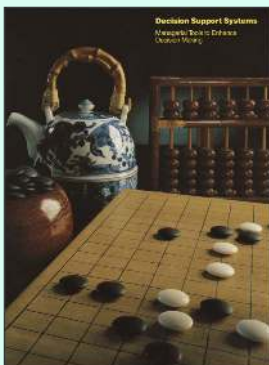
<http://www.economist.com/node/15016132>

Operational issue

what about the long term?

If OLTP systems are perfect for daily management, directors need to know where they should take the business: what is on a downwards trend, which part is picking up, where to invest, and where should the company be in 5 years time. They like to think of themselves as "visionary", they aren't always, but when they aren't having some synthetic ideas about the present state helps (assuming nothing disruptive happens)

DECISION SUPPORT SYSTEMS



It has long been a promise of IT. Some marketing leaflets that are 40 years old could have been written yesterday.

<http://www.computerhistory.org/>

1977 Brochure (Tymshare)


Here the needs of the information system are different. To see long-term trends we need long term data. We don't need every detail of what your latest shopping cart contained. Additional data (how is the competition doing?) helps.

Longer Term

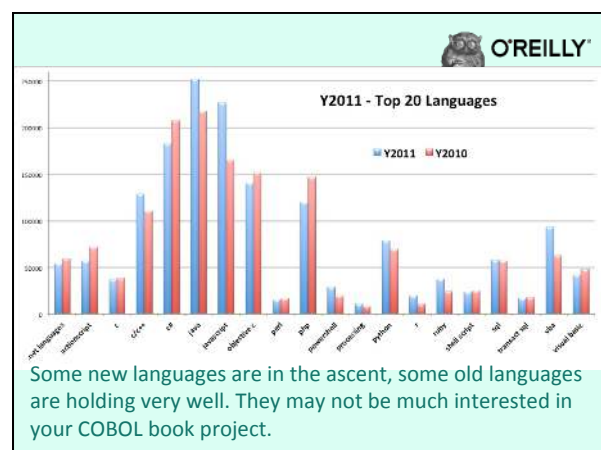
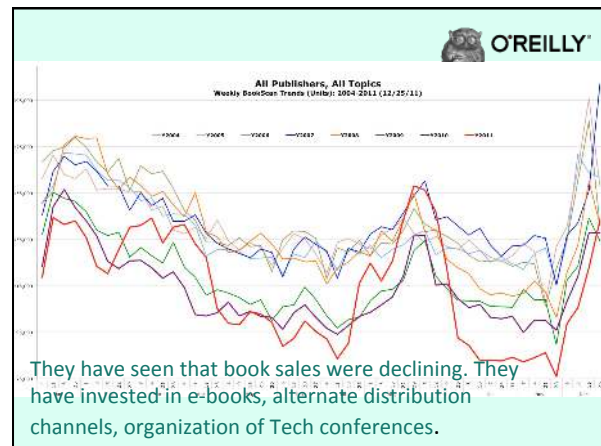
Different Grain of Information

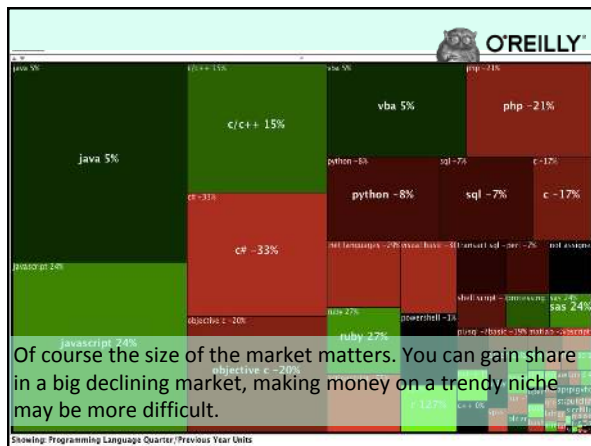
Additional Data

Example



A good illustration is what O'Reilly, the computer book publisher, has been doing for quite a while with data bought from Nielsen, a market research company that compiles book sales (the information here is a few years old, O'Reilly no longer publish it ...)





Never forget that assuming that the future will follow exactly in the steps of the present and the past is always dangerous.

“ Mutual Funds, Past Performance

This year's top-performing mutual funds aren't necessarily going to be next year's best performers. It's not uncommon for a fund to have better-than-average performance one year and mediocre or below-average performance the following year. That's why the SEC requires funds to tell investors that **a fund's past performance does not necessarily predict future results.**

<http://www.sec.gov/answers/mperf.htm>

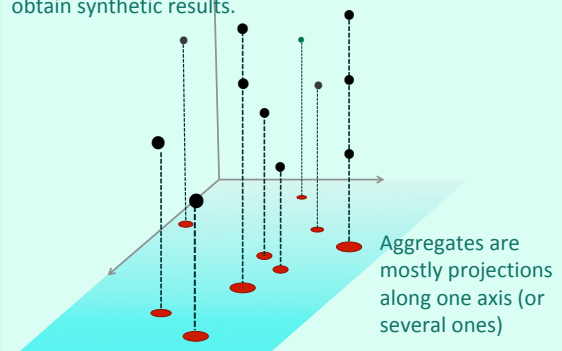
Having longer term data (historical data) and additional data means that we'll have to deal with possibly far more data than operational databases.

Longer Term
Additional Data } **BIG**

What will be big may not be so much what we STORE, if we aggregate data, but what we PROCESS, because aggregation is a costly operation.

Different Grain of Information
Aggregate

And indeed most operations against a data warehouse, and not only its population, will be aggregate operations to obtain synthetic results.

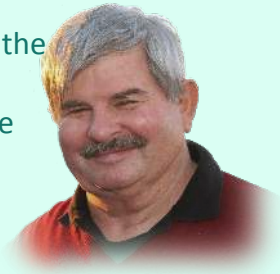


Aggregates, unfortunately, require by definition processing huge volumes of data, sorts, and computations all along. All operations that require I/Os and CPU. The workload is completely different from an operational database workload and this is why data warehousing and business intelligence are considered a topic apart in information systems.

Aggregating takes time

THE PRINCIPLES

Bill Inmon coined the expression "Data Warehouse" in the early 1990s



Bill Inmon

What Inmon had noticed, which has indeed been a recurrent phenomenon in IT since the introduction of personal computers, is that because databases are huge, schemas are complicated and queries difficult to write, a large number of people, especially in higher circles, were mostly working with figures extracted from various sources (including printed reports), and that a lot of strategic data was living outside "official IT"

Shadow IT
Official IT

This shadow IT mostly lives in spreadsheets, shared by email or (today) in a public cloud. Needless to say, data organization is poor, consistency weak, several versions circulate, and security is low.



data spaghetti



not a single version of the truth

Inmon basically suggested creating a single repository that would act as a reference and could be managed with professional (IT) standards.

Inmon's definition

a subject oriented, "themes"
 nonvolatile, Read only
 integrated, Clean
 time variant Historical data
 collection of data in support of

management's decisions

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)

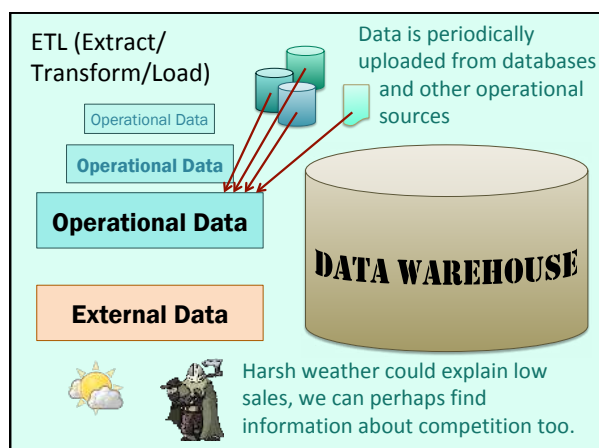
Peculiarities

Read only *we can index happily*

Disposable queries *we can hard-code happily.*

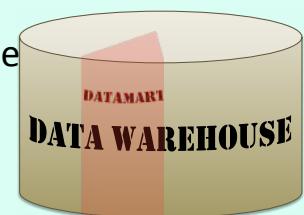


HISTOGRAMS



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



What is important, and this is partly what allows to have a "single version of the truth", is that we aren't interested at all in "real time" data. Availability requirements will also be low compared to operational databases. Dataware House uploads are scheduled on a regular basis, every night, every week-end, or whatever your needs are.

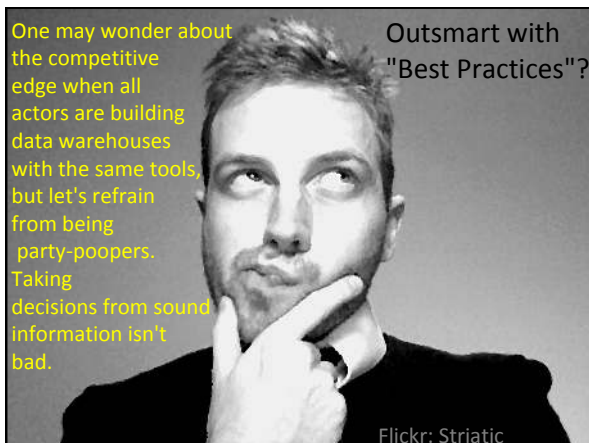
Not "real time" data Uploading schedule

Don't underestimate "Extract/Transform/Load" (ETL) processes. Extraction mustn't interfere too much with operations. Transformation and loading may be complicated (it has spawned an entire industry of tools)

The purpose

To outsmart competition

The promise, of course (and the justification to huge investments in hardware, software licences and consulting) is to give a company that definitive edge over its competitors.



If SQL was initially designed as a language for non-specialists, this is the area where it has most obviously failed. You need tools (IBM Cognos and SAP Business Objects are among the better known - if you feel that the names look like acquisitions, you are right)

The problem

Generating efficient queries for people who cannot spell SQL

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.



Ralph Kimball

2NF

Dimensions

Kimball boldly suggested to shake the dogma of proper database design for decision support systems.

Every non key attribute must provide a fact about the key, 1NF
the whole key, 2NF
and nothing but the key. 3NF

William Kent (1936 – 2005)



2NF Some columns store data derived from other columns

Of course, Kimball knows perfectly what proper normalization means and doesn't discuss the fact that operational systems must be properly normalized. But he points out that the rule that there should be no repeated data in different rows stems from the desire to avoid anomalies when data is updated in some rows and not the others, and that this concern is irrelevant in a read-only database. By contrast, repeating data makes querying easier by removing the need for joins or functions.

To illustrate Kimball's vision, this is what the MOVIES table could look like in a kimballesque database. The English title avoids a join with ALT_TITLES, country name and continent avoid a join with COUNTRIES, storing the decade avoids deriving it from the year.

movieid	movieid
title	title
country	english_title
year_release	country
	country_name
	continent
	year_release
	decade

The feuding brothers of data warehousing



THE CONCEPTS

Mostly Kimball's

To support his ideas, Kimball has introduced a number of new concepts which have become very popular and that you are almost certain to encounter sooner or later.

1

Facts and dimensions

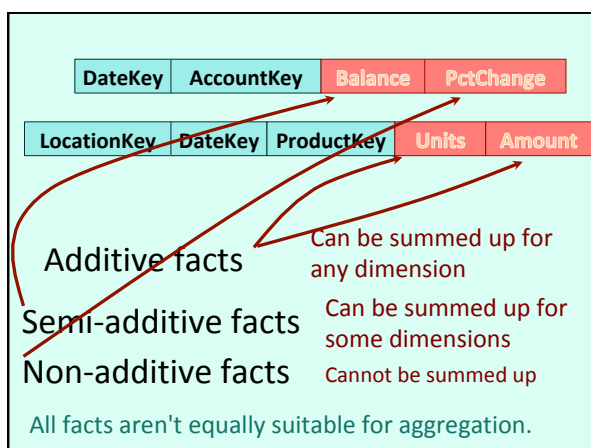
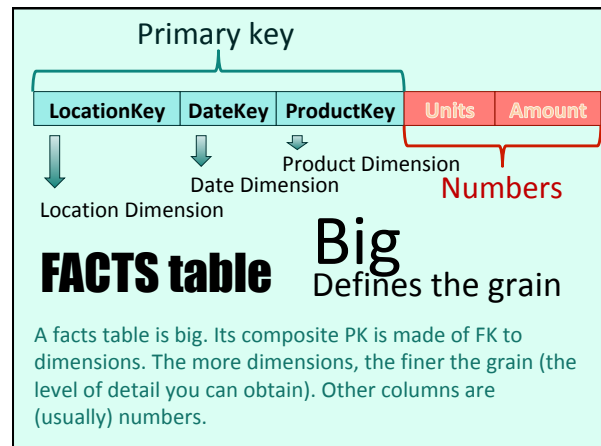
First of all, facts and dimensions. Facts, to make it simple, are principally figures. Dimensions are the axes along which these figures can be aggregated (as we are in a world where we want synthesis). Let's give an example.



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



DateKey	19700101
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	1970
YearMonth	197001
FiscalMonth	7
FiscalQuarter	3
FiscalYear	1970
LastDayInMonthFlag	n
SameDayYearAgo	01/01/69

Dimension table

Date dimensions illustrate perfectly the Kimball approach.

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 deterministic. 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.

See Guy's presentation in a separate document.



Guy Harrison

