Three Things you don’t want to store in a database (but sometimes have to)

Gianluca Sartori

Founder at Quantumdatis

Data Platform MVP

Works with SQL Server since version 7

DBA @ Scuderia Ferrari

Blog: spaghettidba.com
Twitter: @spaghettidba

Agenda

- What are the rules?
- Logs
- Pictures
- Dynamic Attributes

The problem with Normal Forms

The key, the whole key, nothing but the key

- 1NF:
  A primary key, atomic attributes only
- 2NF:
  Every attribute depends on the whole key
- 3NF:
  Every attribute depends only on the key

Bad idea #1
Storing logs in the database

What do you need to do with logs?

- Write efficiently
  - Writing doesn’t have to affect the application performance
  - Written synchronously
  - Mostly written and never read
- Query & analyze
  - You read logs for troubleshooting
  - Usually read sequentially
  - Analysis
- Apply retention policies
  - Keep for a reasonable time, then delete

Logs don’t belong in a database

- It’s not really 1NF compliant
  - You don’t really care about the whole message, but parts of it
  - Huge queries WHERE message LIKE ‘%something%’
- It’s not transactional data
  - You don’t need logs to be ACID compliant
  - When you delete old logs the database log will explode
  - You need to log when the database is not reachable
- There is no benefit
  - RDBMSs deal with relational data very well
  - Logs are… logs. What can a RDMBS offer?

What are the alternatives?

- Files
  - Easy to write to
    - Performance is great
    - Available when the database is offline / unreachable
  - Easy to read
    - Text editors & low tech
    - Human readable
  - Easy to query
    - GREP
    - LogParser
  - Easy to delete
    - Just delete old files

What are the alternatives?

- Specialized logging databases
  - Logstash
  - Loki
  - Loggly
  - ElasticSearch
  - InfluxDB
  - GrayLog
  - DataDog
  - Splunk
- Strengths:
  - Optimized for time
  - Analysis

Why in a RDBMs?

- Easy - lazy
  - Not different from any other type of data
  - No need to learn new techniques / technologies
- Convenient for filtering & displaying data
  - Filter with SQL queries
- Can leverage RDBMS features
  - High Availability
  - Replication
  - Backups

Challenges

- High cardinality
  - Storage
  - Querying
  - Retention

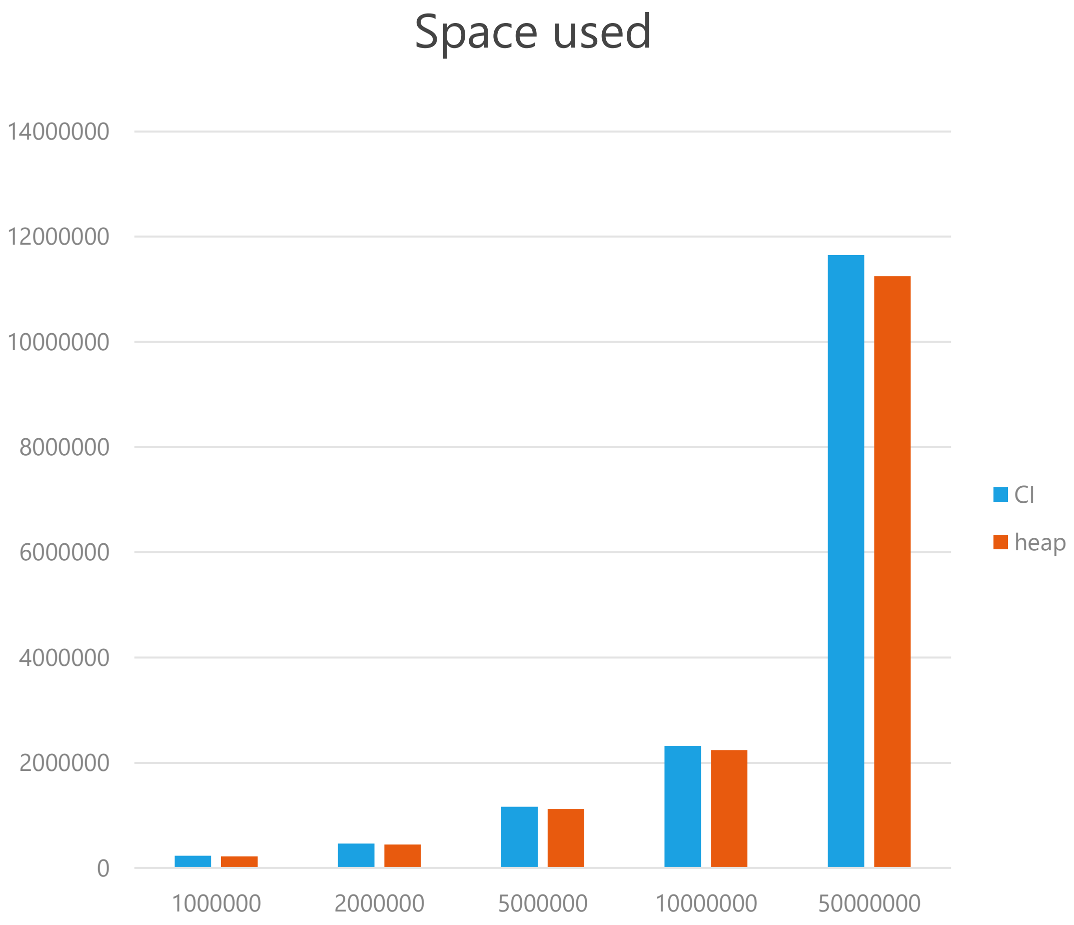
Possible solutions

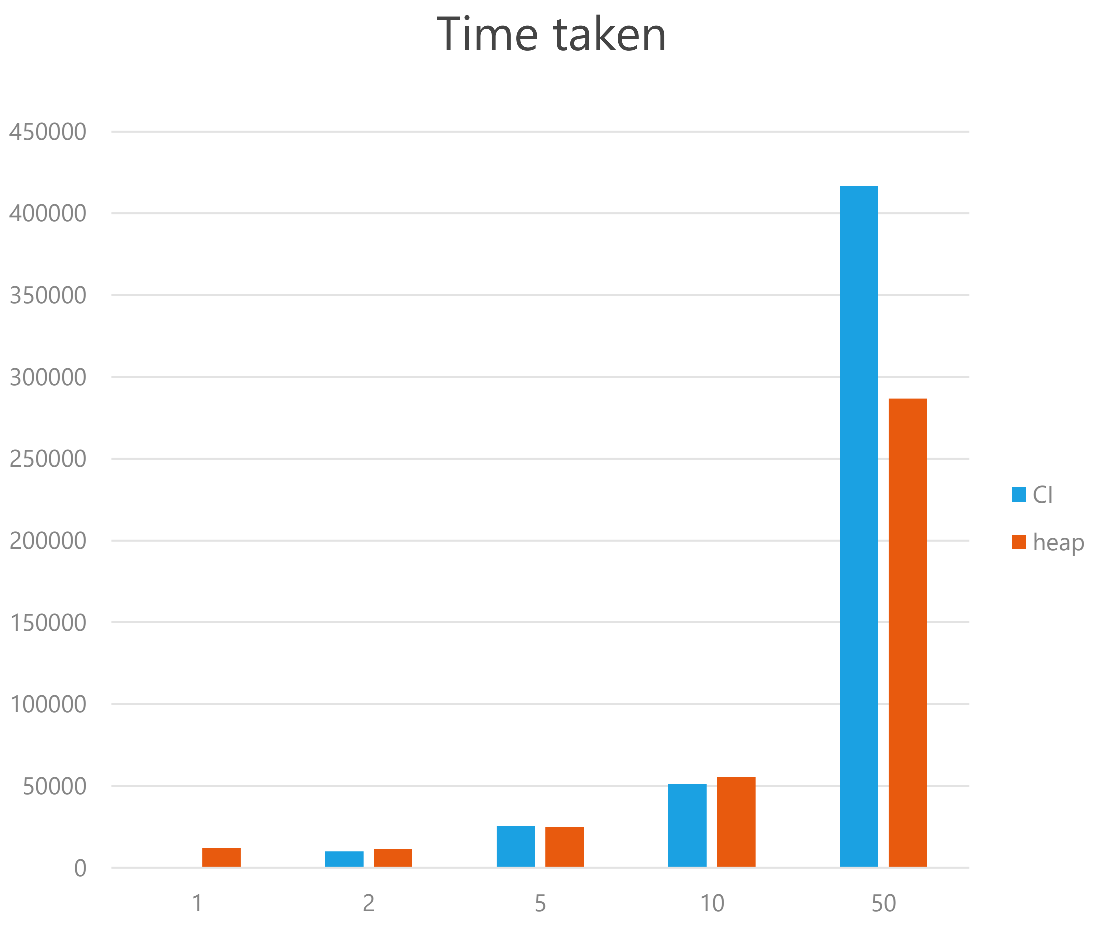
- Choosing the right storage
  - Heaps?
  - Clustered Indexes?
  - Compression?
  - Columnstore indexes?
  - Partitioning?

DEMO

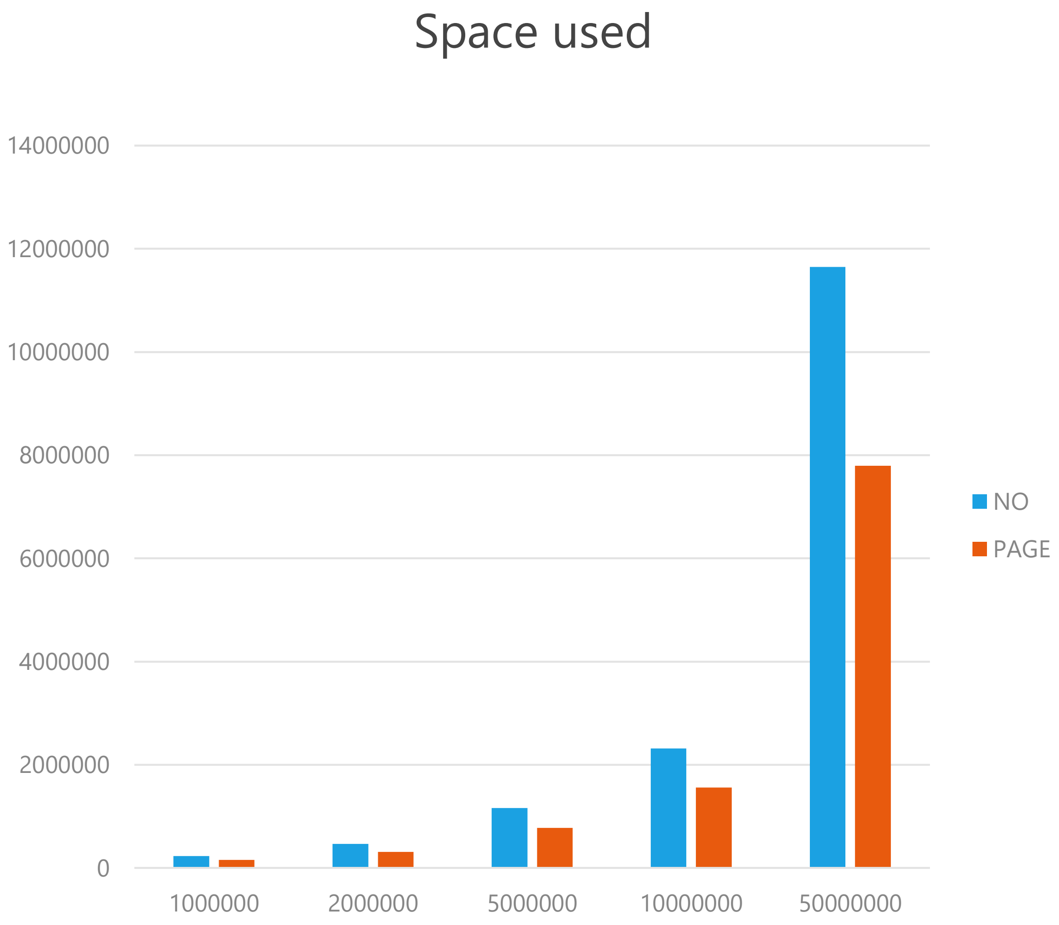
LOGS

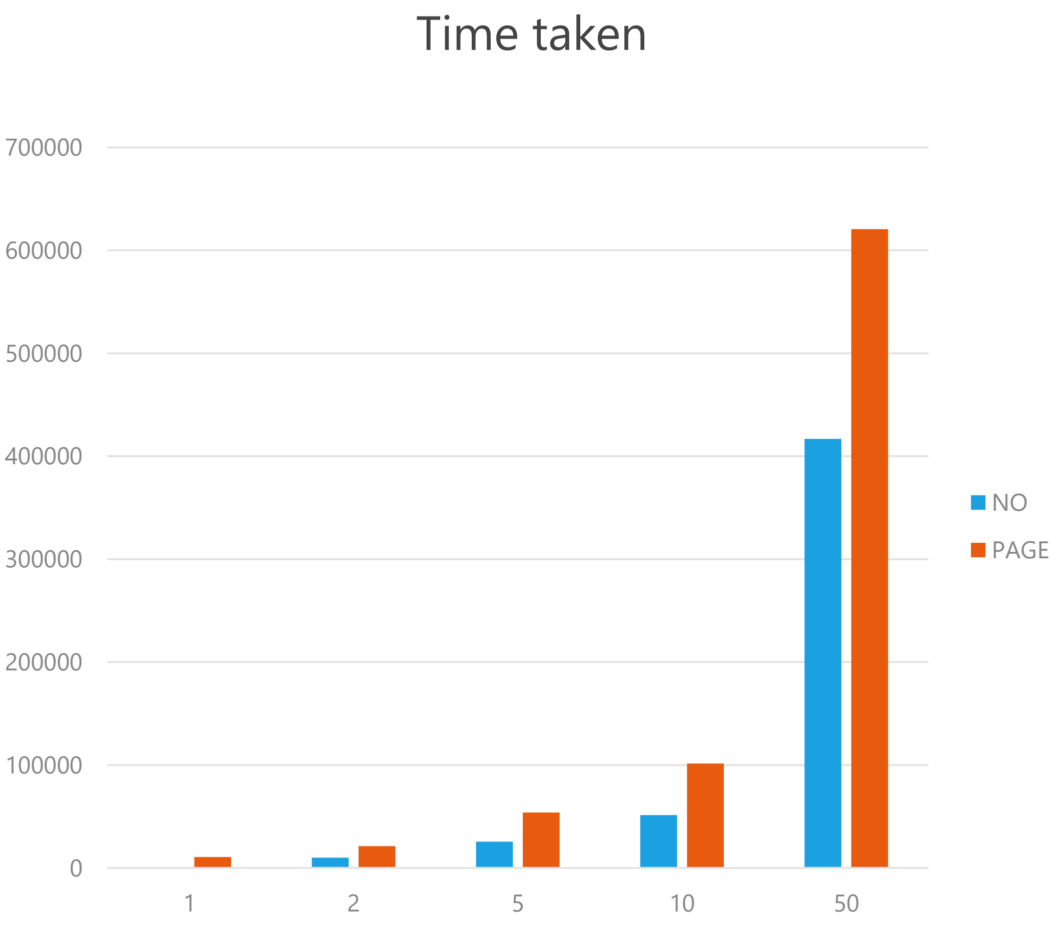
Heaps vs Clustered Indexes



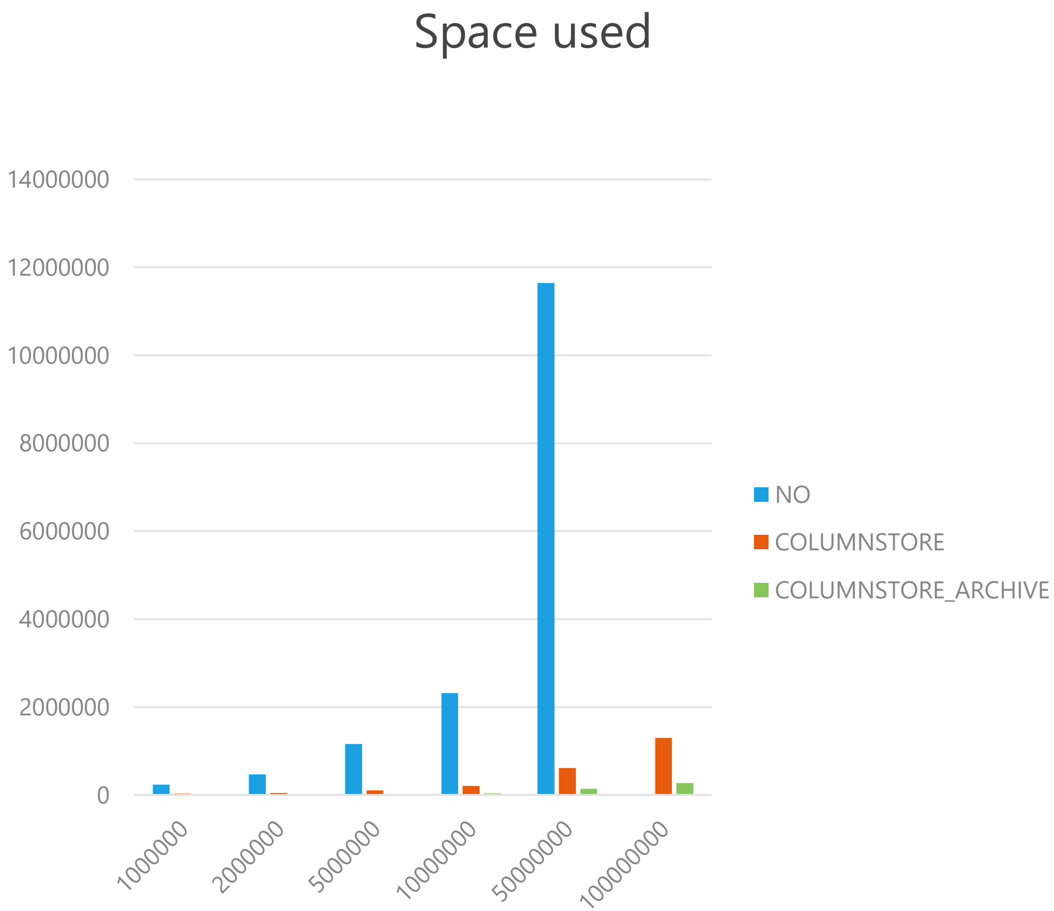


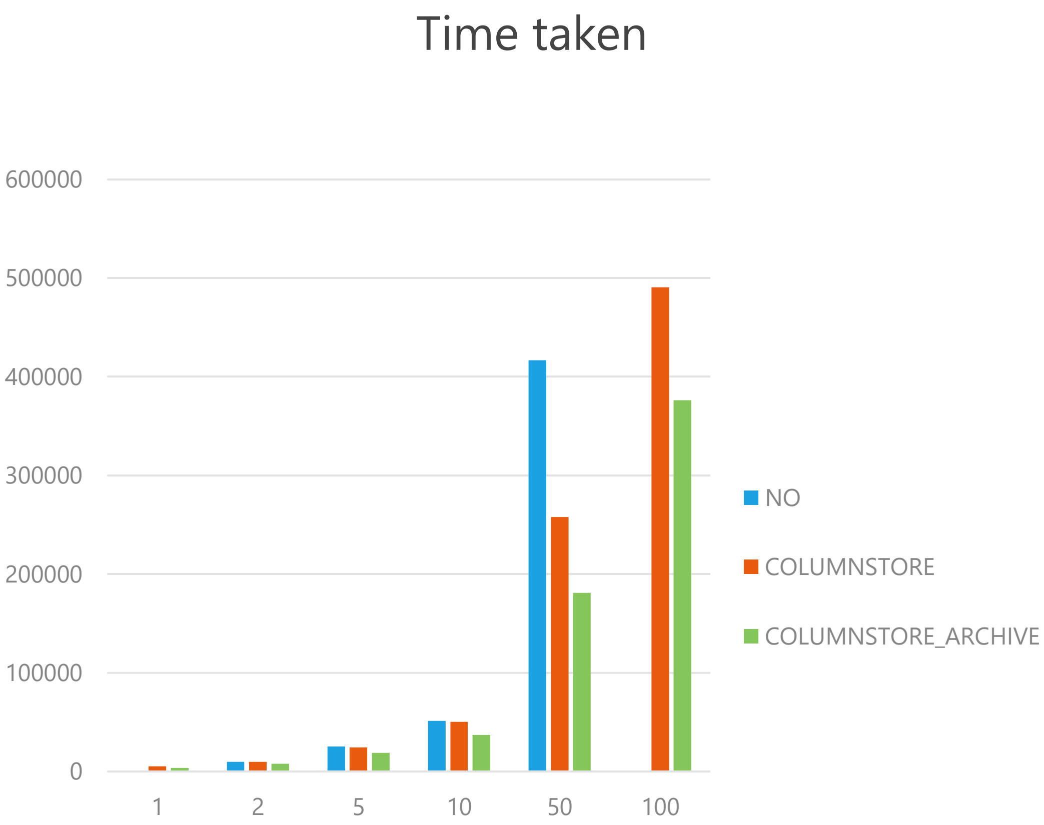
Compressed vs Uncompressed





Columnstore vs Rowstore





Wrap up

- Columnstore wins because of the shape of the data
- If logs are not repetitive enough it might be a problem
- Careful with LOB data
- Use partitioning to compress rows
- Use partitioning to delete old data
- Don’t try to delete rows on compressed segments: columnstore doesn’t like deletes

Bad idea #2
Storing pictures in the database

What do you need to do with pictures?

- It’s not only pictures, it’s binary files in general. Pictures is a very common example

- Write once, read as a whole
  - Mostly read, seldom written
  - Always read entirely
- Pass to other applications
  - Read the contents, pass to a web server or client app
  - Always read and write entirely

Pictures don’t belong in a database

- It’s not transactional data
  - You don’t need pictures to be ACID compliant
  - When you maintain indexes the database log will explode

- There is no gain, only pain
  - RDBMSs are designed for relational data
  - Pictures don’t fit well in pages/extents
  - Throughput is poor

What are the alternatives?

- Files
  - Easy to write to.
    - Performance is great
    - Available when the database is offline / unreachable
  - Easy to visualize/edit
    - Low tech
  - Easy to update/delete
    - Just replace or delete files

What are the alternatives?

- Specialized object storage
  - Azure BLOB storage
  - Amazon S3
  - … all cloud providers have a solution
  - NetApp
- Strengths:
  - Optimized for analytics
  - History
  - Distributed
  - Scale-out

Microsoft Azure
Blob Storage

amazon
S3

Why in a RDBMs?

- Easy - lazy
  - Treat as the rest of the data
  - No need to learn new techniques / technologies
- Can leverage RDBMS features
  - High Availability
  - Replication
  - Backups

Challenges

- Scales very poorly
  - Reads and writes are poor compared to file system
  - Index maintenance is virtually impossible
  - Contributes to backup size

Possible solutions

- Choosing the right storage
  - Heaps?
  - Clustered Indexes?
  - Compression?
  - Columnstore indexes?
  - Filestream

DEMO

PICTURES

Wrap up

- Filestream delivers best performance
- Be careful with missing files (CHECKDB will detect)
- NTFS performance affects filesystem access
    - Disable 8.3 naming
    - Disable indexing
    - Disable lastaccess
    - Dedicated disk
    - Choose correct cluster size
    - Exclusions for antivirus / antimalware
- Skipping TDS might improve performance

Bad idea #3
Dynamic attributes in the database

Why dynamic attributes?

- Sometimes it’s hard to identify all attributes at design time
- Some applications may have the ability to store dynamic attributes
  - CRM
  - RAD

Dynamic attributes don’t belong in a database

- All attributes of entities must be known at design time
- The relational model does requires equal number of attributes for each row
- Some (bad) implementations violate 1NF, 2NF or 3NF

What are the alternatives?

- Document databases
  - Azure Cosmos DB
  - MongoDB
  - Amazon DocumentDB
  - ElasticSearch

Amazon DocumentDB

Azure
Cosmos DB

mongoDB®

elasticsearch

Why in a RDBMs?

- Looks «smart»
  - There are many possible implementations (some are very bad)
  - Stays with the rest of the data
- Can leverage RDBMS features
  - High Availability
  - Replication
  - Backups

Possible solutions

- EAV
  - Violates 2NF, 3NF
- XML
  - Violates 1NF
- JSON
  - Violates 1NF
- Sparse columns
  - Lots of NULLs
  - Generic attribute names

DEMO

DYNAMIC ATTRIBUTES

Wrap up

- Dynamic attributes have no place in relational databases
- XML and JSON violate 1NF – square peg, round hole
- EAV is the worst
  - Generic data types → Eg: varchar(4000)
  - No Foreign Keys
  - No CHECK constraints
  - Multiple accesses to the same table

- SPARSE columns are not too bad
  - Doesn’t work when every row has completely different attributes

Wrap up

To store data in a database, you need to know the rules.
In a relational database, the rules are called Normal Forms.

Ignore them to enter a world of pain.