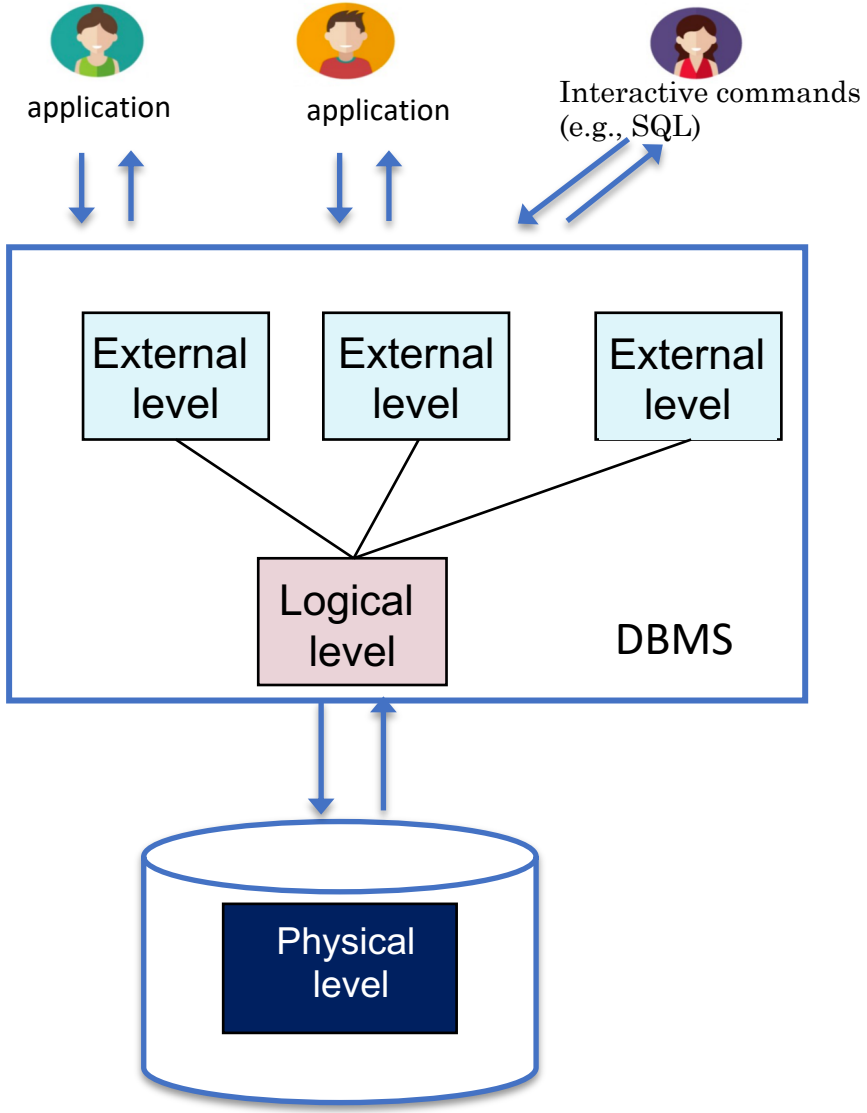
Aggregate-oriented NoSQL data stores

Introduction

Data models

- A data model is a set of constructs for representing a certain reference domain
- Logical model: how entities and their associations are represented inside the system
  - Relational model: tables, columns and rows
- Storage model: how the DBMS stores and manipulates the data at the physical level
- A logical model is usually independent of the storage model
- In NoSQL systems, this is no more true

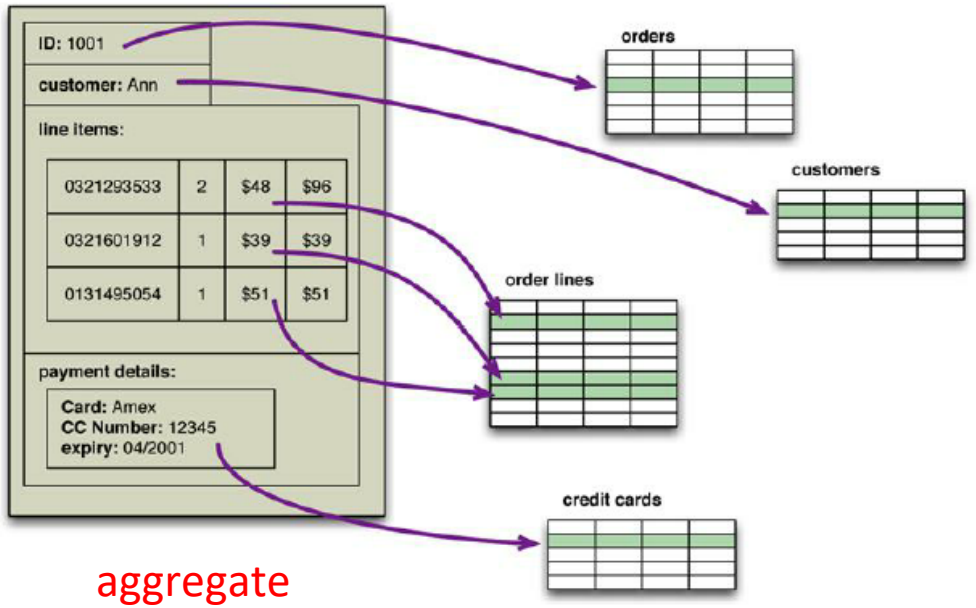


Aggregate: definition

- **Logical level**: an aggregate is a data unit with a complex structure

- Not simply a tuple (a table row) like in RDBMS
  - Example: complex record with: simple fields, arrays, records nested inside

- For data manipulation and management of consistency

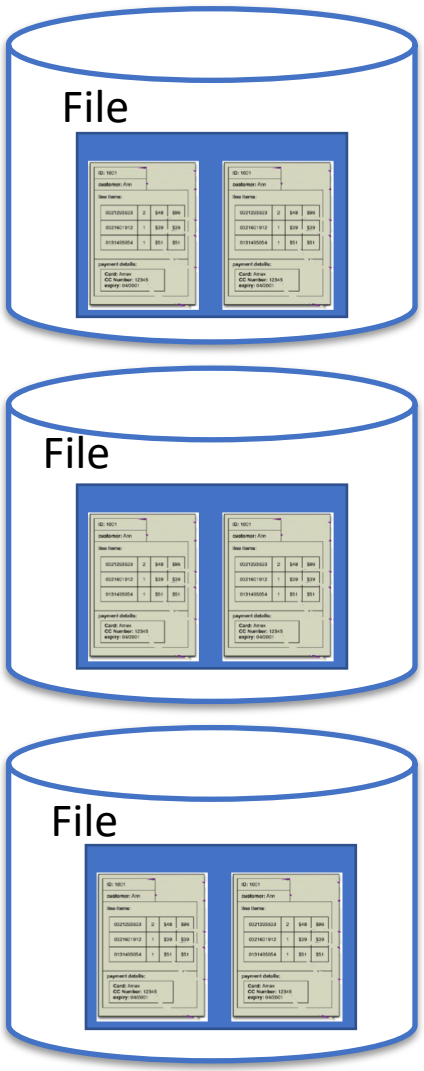


aggregate

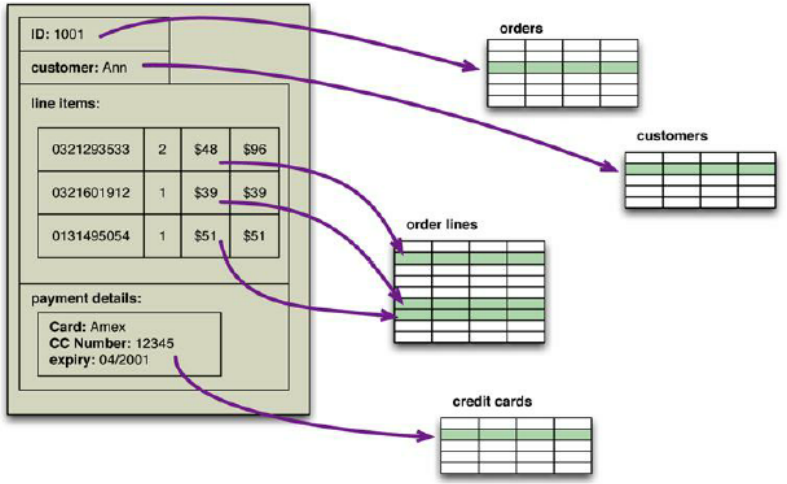
relational schema + instance

Aggregate: definition

- **Physical level:** an aggregate is the unit of interaction with the data store
- All the data about a unit of interest (an aggregate) are kept together on the same node
- Partitioning separates different units (different aggregates) on different nodes



Aggregates

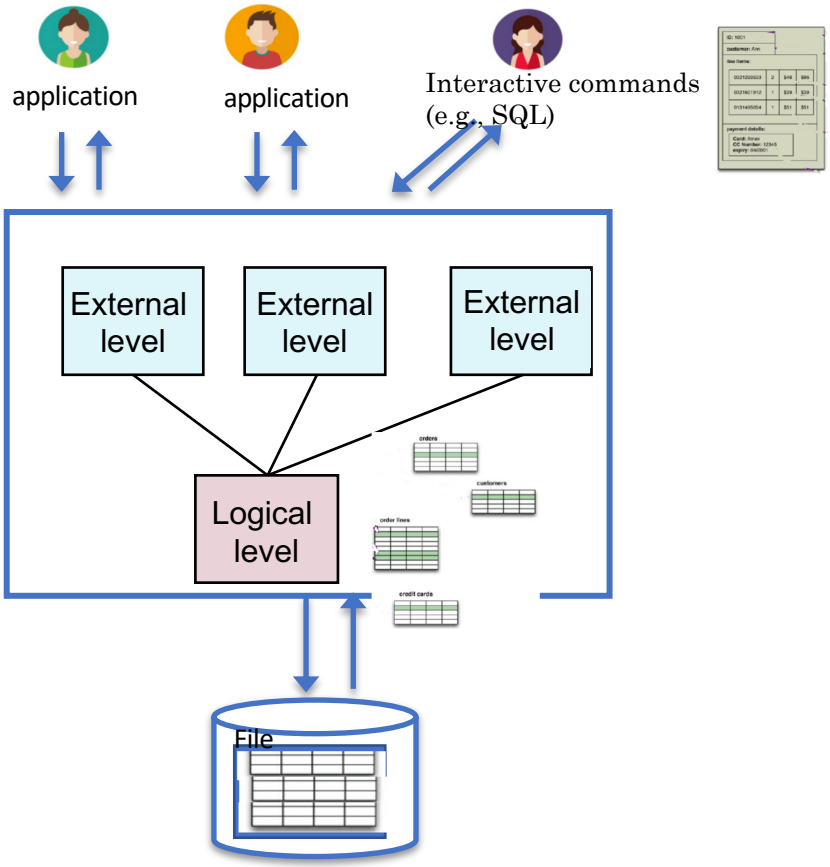


- Relational model is aggregate-ignorant
  - It is not a bad thing, it is a feature
  - Allows to easily look at the data in different ways
  - Best choice for integration database (no primary structure for data manipulation)

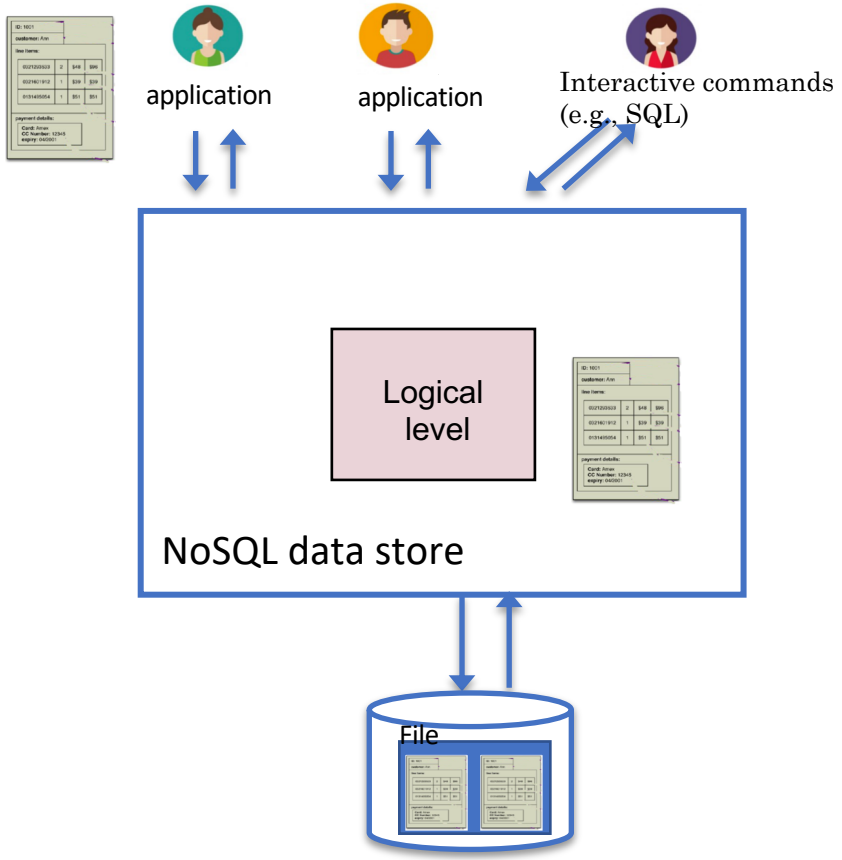
- Advantages of aggregates:
  - good option for application database
  - easier for application programmers to work with
  - easier for database systems to handle operating on a cluster
  - limit impedence mismatch: strict relationship with JSON, a lightweight format for data exchange

JSON: see short introduction on Aulaweb: Resources

Aggregates



Integration DB
SQL-based system

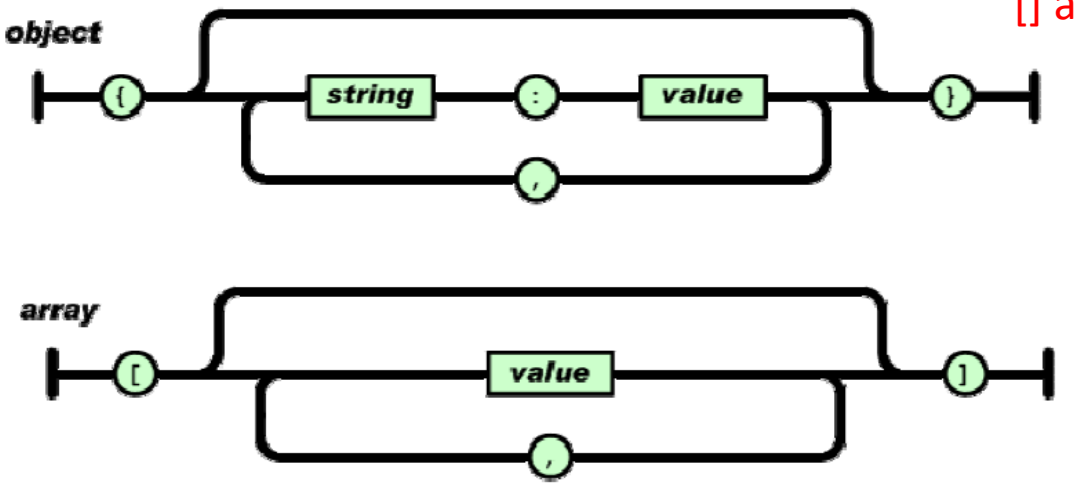


Application DB
NoSQL aggregation-dependent system

JSON

- In case you haven't seen it, here is a quick introduction to JavaScript Object Notation

{} object (name-value pair)
[] array



- JSON schema can be used to specify the structure of a set of JSON objects
  - Semistructured schema

Example

- Imagine we decide to build a product review database, collecting together all the reviews of a given product:

Product: iPhone 12
Reviewer: 458743
Date: 12.4.2021
Price: 1.100
Camera: Fine
Screen: Very good
Accessories: {Headphone, Case, ….}

Product: iPhone 12
Reviewer: 636534
Date: 30.5.2021
Price: 1.000
Camera: Excellent
Screen: Poor in sunshine
Operating system: Easy to use

- Different products have different fields
- Some fields have several values
- Products are related to other products as accessories
- A new product may have new qualities (fields) not yet in the database
- The same product may appear many times with different values and fields

Example JSON

{} object (name-value pair)
[] array

Product: iPhone 12
Reviewer: 458743
Date: 12.4.2021
Price: 1.100
Camera: Fine
Screen: Very good
Accessories: {Headphone, Case, ….}

Product: iPhone 12
Reviewer: 636534
Date: 30.5.2021
Price: 1.000
Camera: Excellent
Screen: Poor in sunshine
Operating system: Easy to use

```
{
"Product": "iPhone 12",
"Review":
        {"reviewer": "458743",
        "date": "12.4.2021",
        "camera": "Fine",
        "screen": "Very good",
        "price": 1.100,
        "accessories": ["Headphone","Case"]}
}
```

```
{
"Product": "iPhone 12",
"Review":
        {"reviewer": " 636534 ",
        "date": "30.5.202",
        "camera": "Excellent",
        "screen": "Poor in sunshine",
        "price": 1.000,
        "operatingSystems": "Easy to use"}
}
```

https://json-schema.hyperjump.io/ for validation

Example JSON schema

```
{
  "type": "object",
  "title": "Review",
  "description": "A review for a certain product",
  "properties": {
            "Product": { "type": "string", "description": " The product for which the review is given" },
            "Review": {
                        "type": "object",
                        "properties": {
                                    "reviewer": { "type": "string", "description": "" },
                                    "date": { "type": "string", "description": "" },
                                    "reviewer": { "price": "string", "description": "" },
                                    "camera": { "type": "string", "description": "" },
                                    "screen": { "type": "string", "description": "" }
                                              }
                      }
                        },
   "required": ["Product"]
}
```

https://json-schema.hyperjump.io/ for validation

Aggregates and JSON

- We rely on JSON and a simplified JSON schema for describing aggregates at a meta-logical level
- Close but not coinciding with logical models provided by aggregate-oriented NoSQL data stores
- Useful for understanding aggregate modeling before implementation

Properties achieved

1. Application database
2. Flexible schema, possibly unstructured data
3. More complex data
4. Joins are an issue, normalization is no more a reference principle
5. Mainly procedural code

Aggregate properties

- Aggregates give the database information about
  - which portions of data will be manipulated together (logical view)
  - what should be stored on the same node (physical view)
  - *strict relationship between logical and physical levels*

- Minimize the number of nodes accessed during a search (*if the aggregate is modeled taking into account the workload*)

- Impact on concurrency control and consistency
  - NoSQL databases typically support atomic manipulation of a single aggregate at a time
  - Update that affects multiple aggregates leaves open a time slot during which clients could perform an inconsistent read
  - Part of the consideration for deciding how to aggregate data

Aggregate-oriented logical data models

- Aggregate-oriented NoSQL databases based on aggregates are categorized according to the characteristics of aggregates themselves [Rick Cattell, 2010]
  - Key value
  - Document-oriented
  - Column family
- Entities are represented as pairs (key, value):
  - key is an identifier (not necessarily unique among a collection) of an entity
  - value describes the entity structure and corresponds to an aggregate, following a JSON style
- Different models differ for the degree of structure associated with aggregates and, as a consequence, the types of manipulations to be applied over aggregates
- At the physical level, the key is the partitioning value: different aggregates associated with the same key are stored in the same node

Aggregate-oriented logical data models