Chapter 3 : Stars & Cubes

Surrogate keys
Natural keys

Sample Star Schema

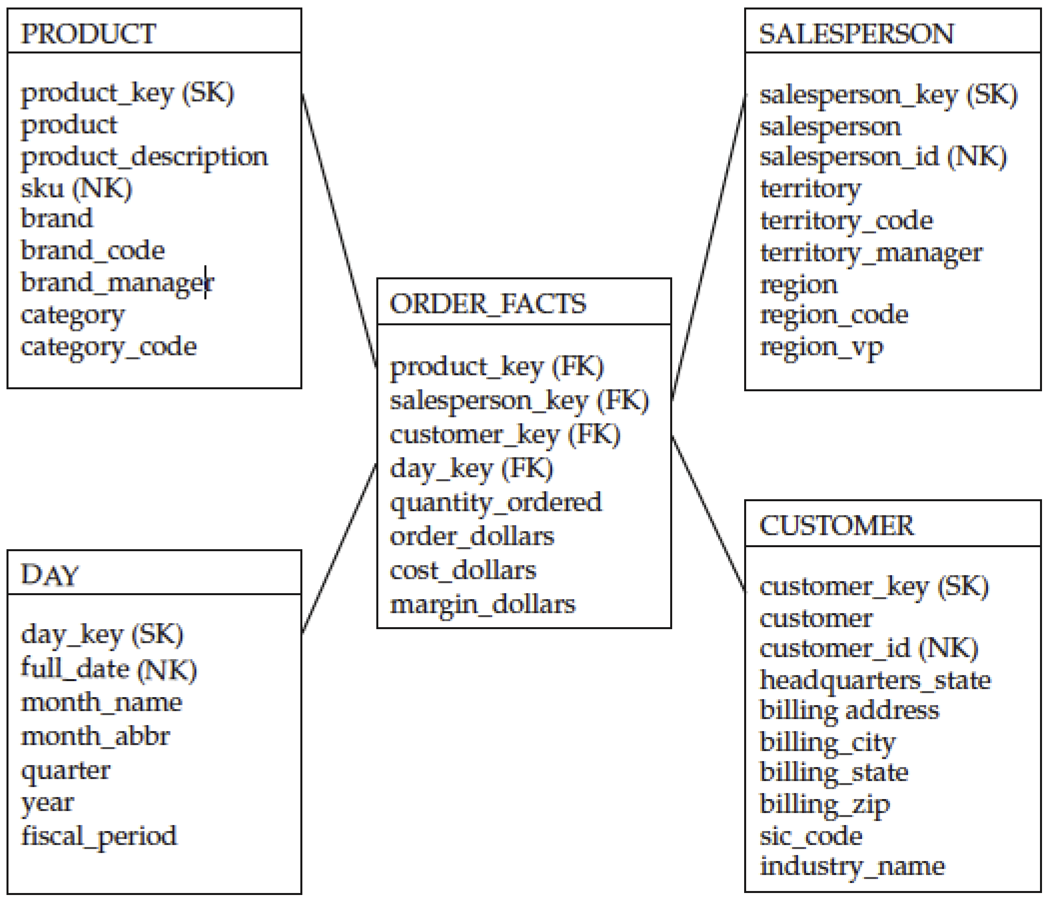


Figure 3-1 Surrogate keys (SKs) and natural keys (NKs)

Surrogate keys

Surrogate keys allow an interesting technique for managing changes in source data

Alternatives:

- Supplement natural key with sequence no
    - Results in complicated FKs, joins … hard to read SQL
- Supplement NK with timestamps
    - Similar issues to above

Rich dimensions

Introduce attributes to simplify querying, filtering, …

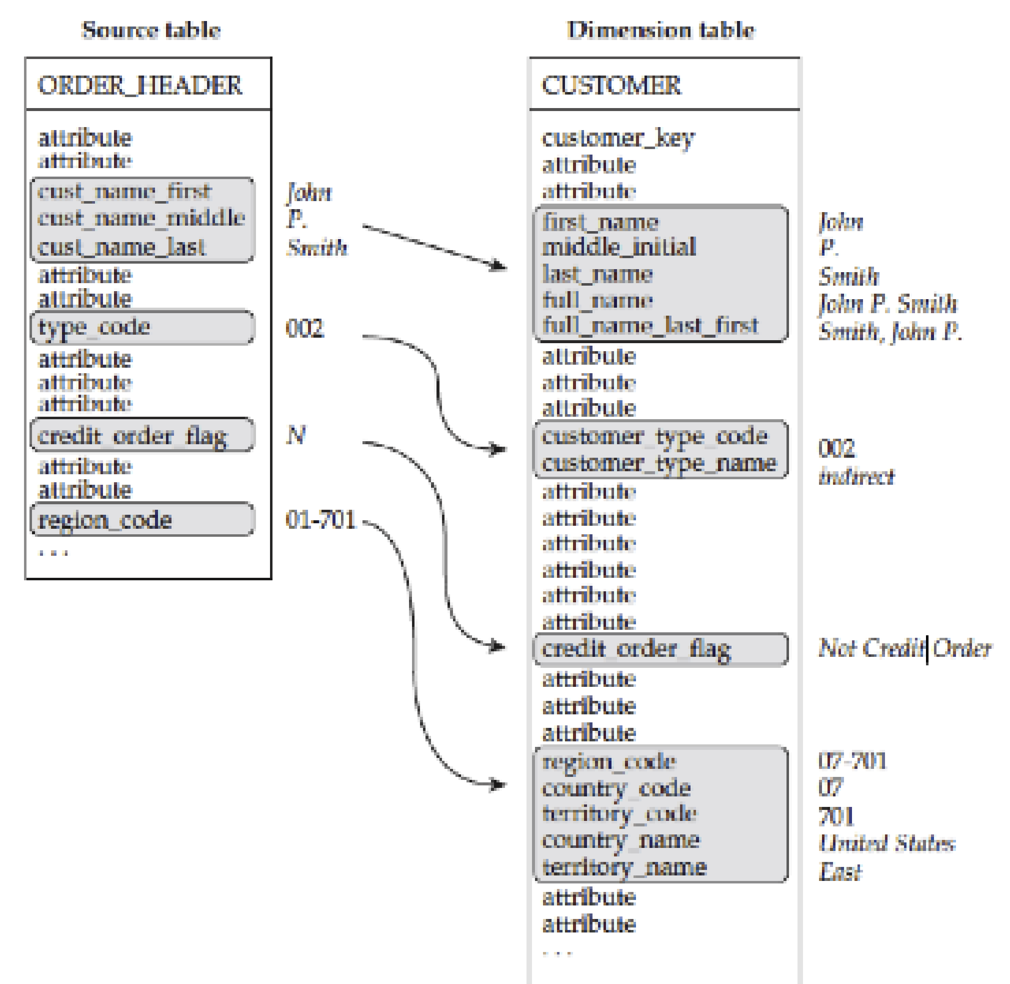


Figure 3-2 Constructing a rich set of dimension attributes

Rich dimensions

Wide tables with lots of attributes with the expectation of:

- •Simplifying query building – less functions, meaningful attribute values, simple design
  - •day of week, am/pm , etc can be stored or derived

- •Speeding up queries due to fewer joins, less derived data
  - •Snowflakes vs denormalized
    - • Snowflakes → normalized dimensions
  - •Codes and descriptions

- •Rigorous analysis of data leads to consistent data across schemas
  - •e.g. codes are synthesized: male/female vs m/f, male/female, 0/1, etc.

Rich dimensions

Common combinations

- Break data element down to component parts – include these
- Include other reasonable combinations to facilitate analysis
- e.g. names
  - First name
  - Middle names
  - Last name
  - First-last
  - Last-comma-first
  - etc

Rich dimensions

Codes & descriptions

•Tables of codes and descriptions exist in operational systems. Referencing tables store the code as a FK into the code table.

•In DM, we include the code & the description in the dimension. Not likely to have a code table (except in ETL tables).

•e.g. address

street
city
provinceCode
Province
…

Rich dimensions

Flags & their meanings

- Flags are commonplace in operational systems.
    - May be boolean, strings ("1","0","true","false",'t","f", …), etc

- In DM, its useful for queries to have the actual value/meaning stored

- e.g. products in Northwind can be discontinued. Instead of a boolean for 'discontinued' we can store "discontinued", "not discontinued"

Rich dimensions

Multi-part fields

•Operational systems often have fields that have multiple components. At UW we could find a field for ‘section’ that contains values like “ACS-4904-001/3”

•In DM, its useful to have the actual value and its component values as separate fields, such as:

Full section number	ACS-4904-001/3
Department	ACS
Course number	4904
Section number	001
Credit hours	3

Rich dimensions

Numeric fields

•Sometimes there's confusion: Should a numeric field be in a dimension, or should it be in a fact table?

•In DM, consider how the field will be used. Is it used to summarize or categorize other metrics? Is it aggregated in reports?

•E.g. quantity ordered: we probably wouldn't need to see the number of times someone ordered 10 of something, rather we're more likely to sum the quantity ordered ➔ fact

Rich dimensions

Numeric fields

- E.g. unit price: by itself not something to summarize, but in conjunction with quantity and discount it is. So its better to place unit values in a dimension and put extended values in a fact table.

- If necessary we can summarize by pulling dimension attributes into a query so nothing is lost.

Grouping attributes into dimensions

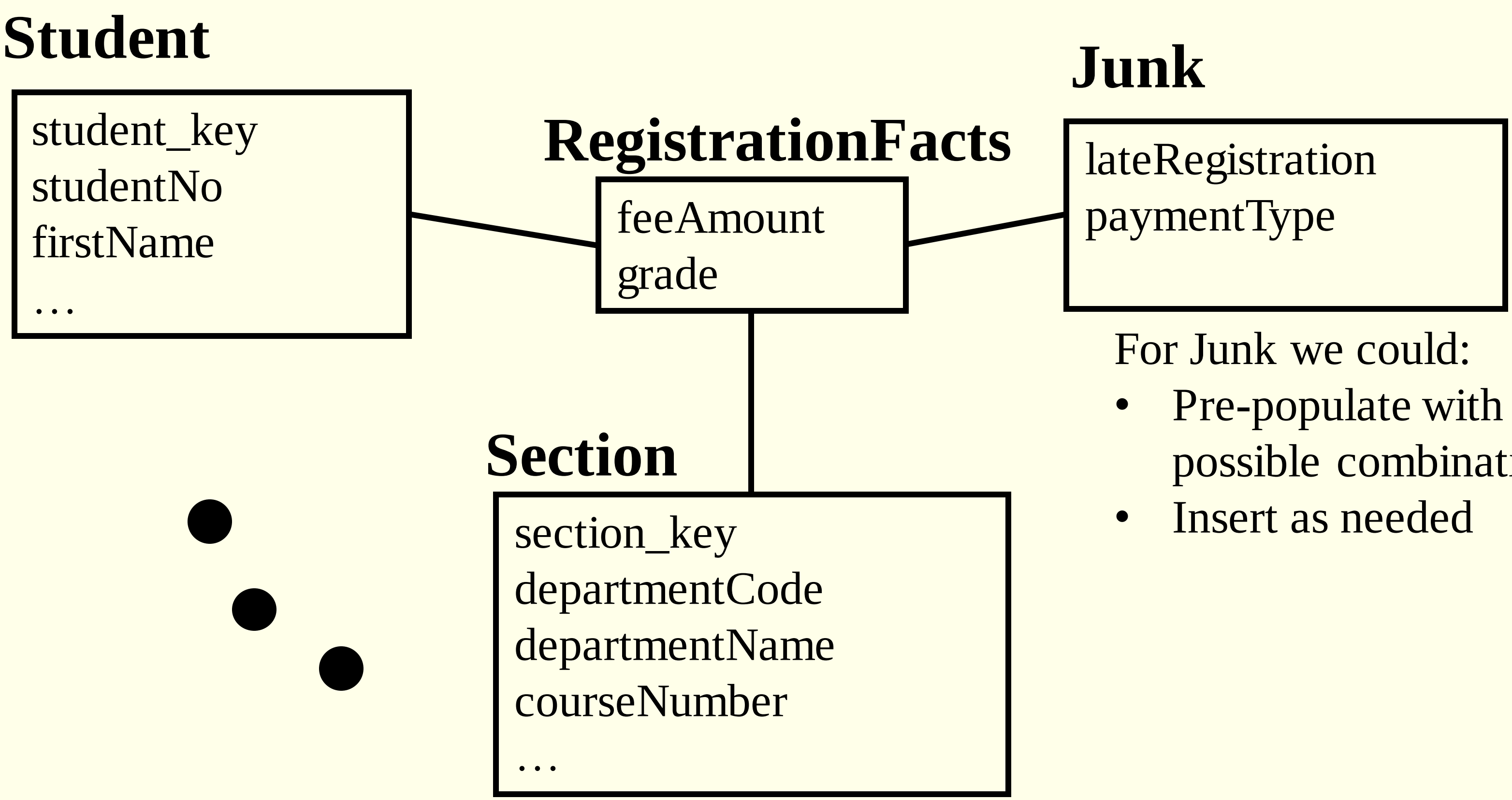
Attributes are grouped into tables representing various entity types.

- E.g. student, course, instructor, department, …

- Junk dimensions
  - Sometimes there may be no place for some attributes, or the grouping is so small, we may wish to combine these into a junk dimension
  - Generally speaking, the grouped attributes have no affinity for each other

Junk dimensions

How do we populate a junk dimension?
E.g: suppose we have a student registration schema with the junk dimension shown..

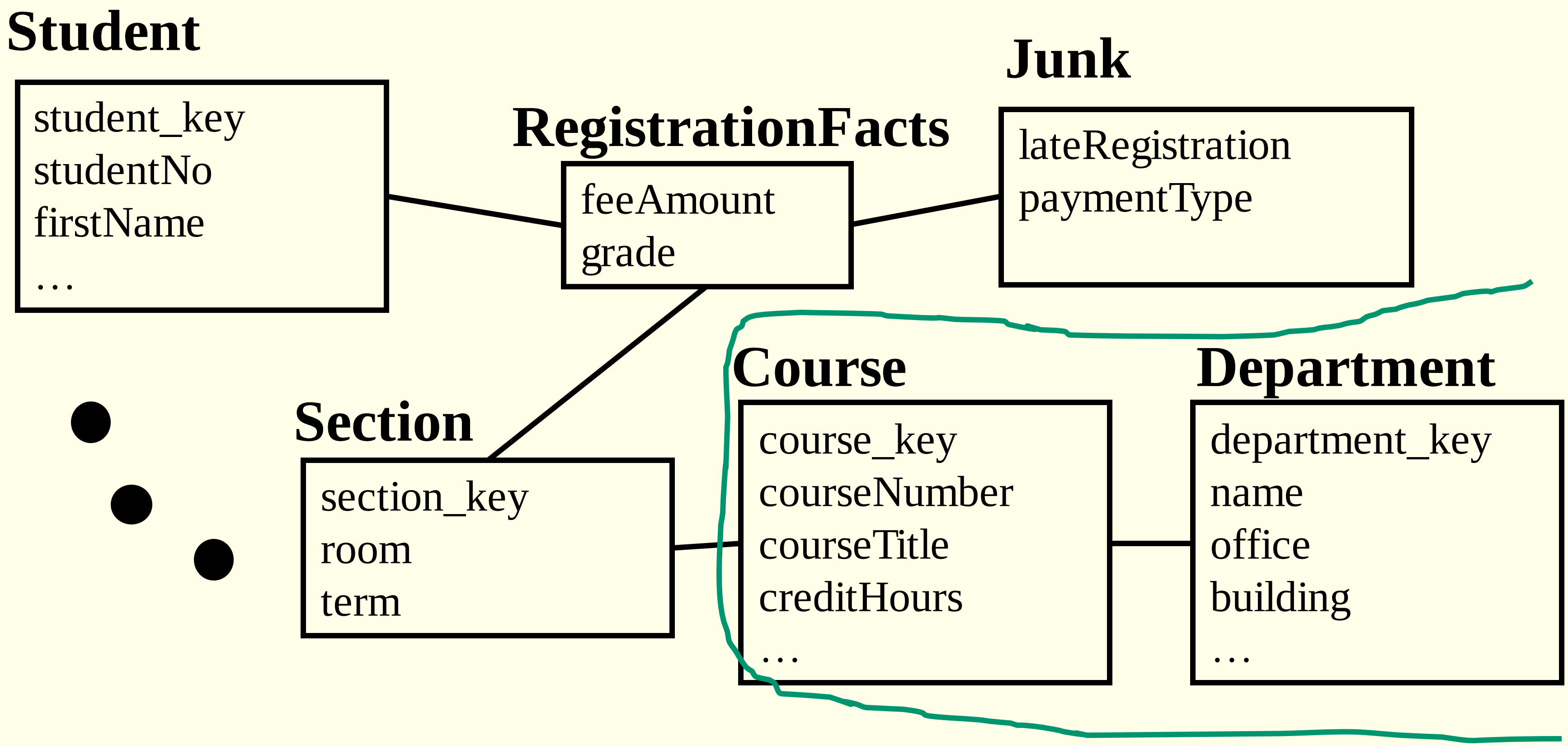


For Junk we could:

- Pre-populate with all possible combinations
- Insert as needed

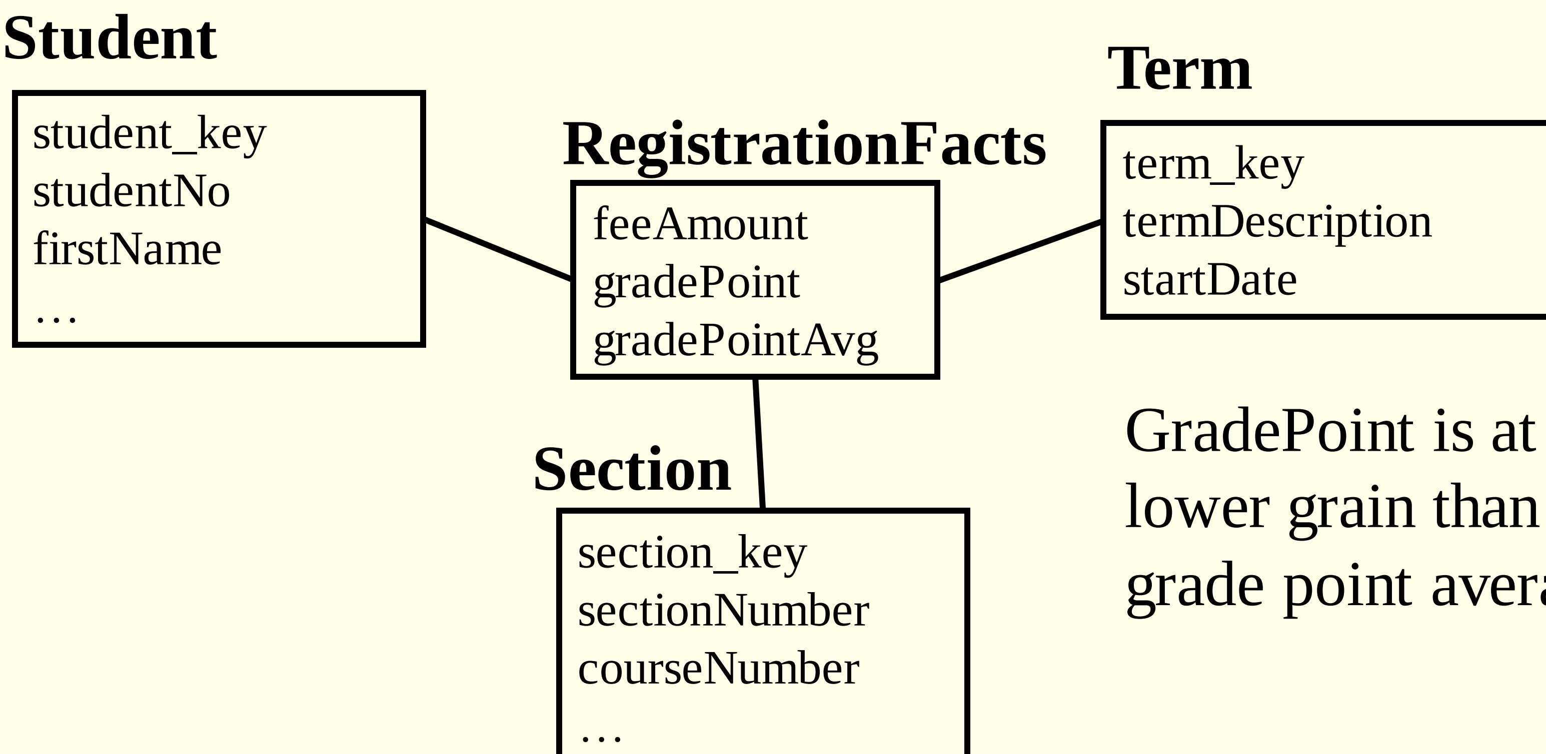
Snowflaking

If we normalize dimensions then we say we have a *snowflake* design where the additional tables are called *outriggers*.



Fact Tables

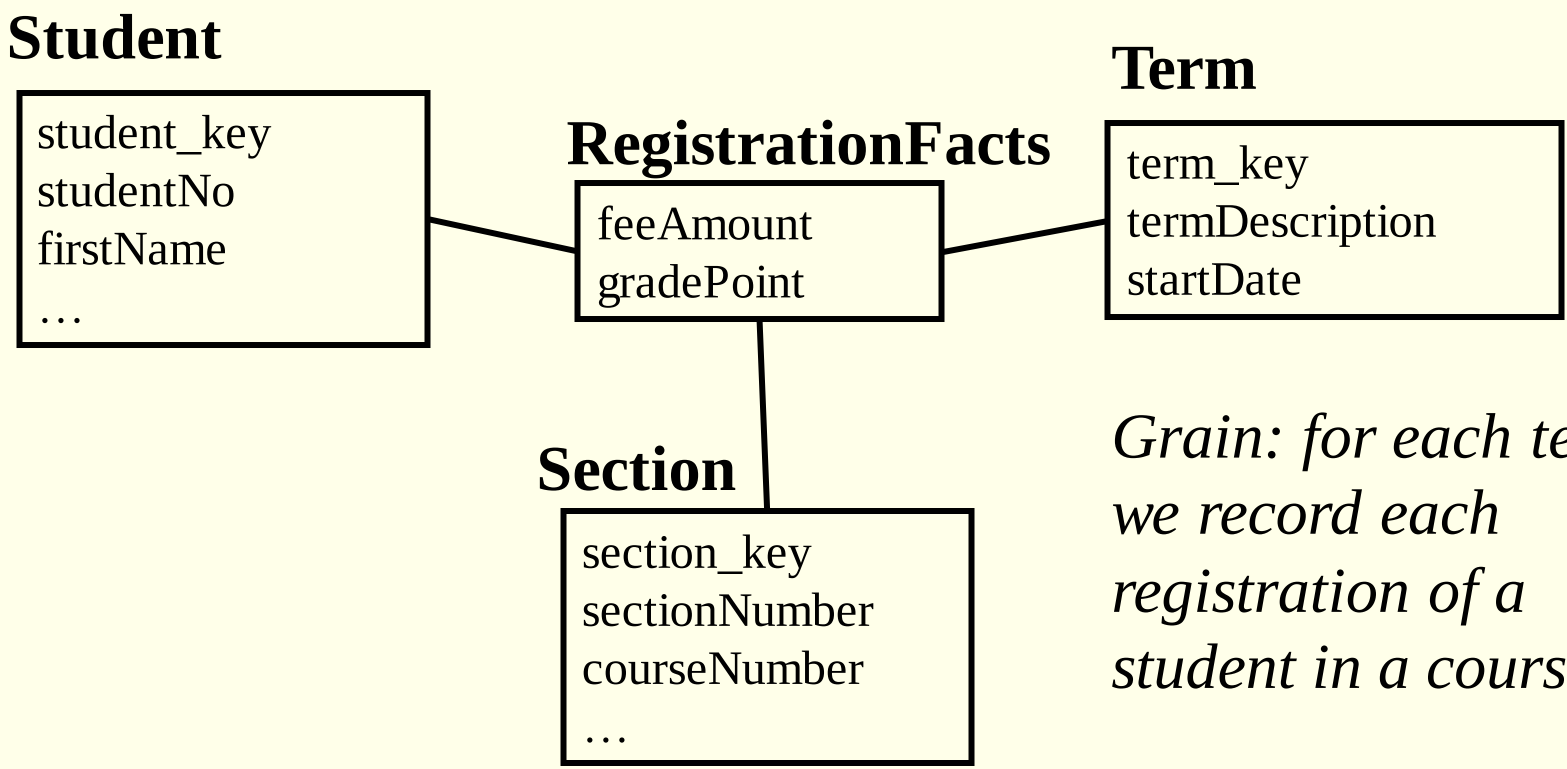
Grain of the fact table is the level of detail it represents.
RoT: The fact table should hold facts **at one grain only**.
E.g. the following schema holds grades and grade point averages.



GradePoint is at a lower grain than grade point averages

Fact Tables - sparse

In general, a fact table does not have a row for every combination of dimension rows.
Below, there is one registration fact for every course taken by a student.

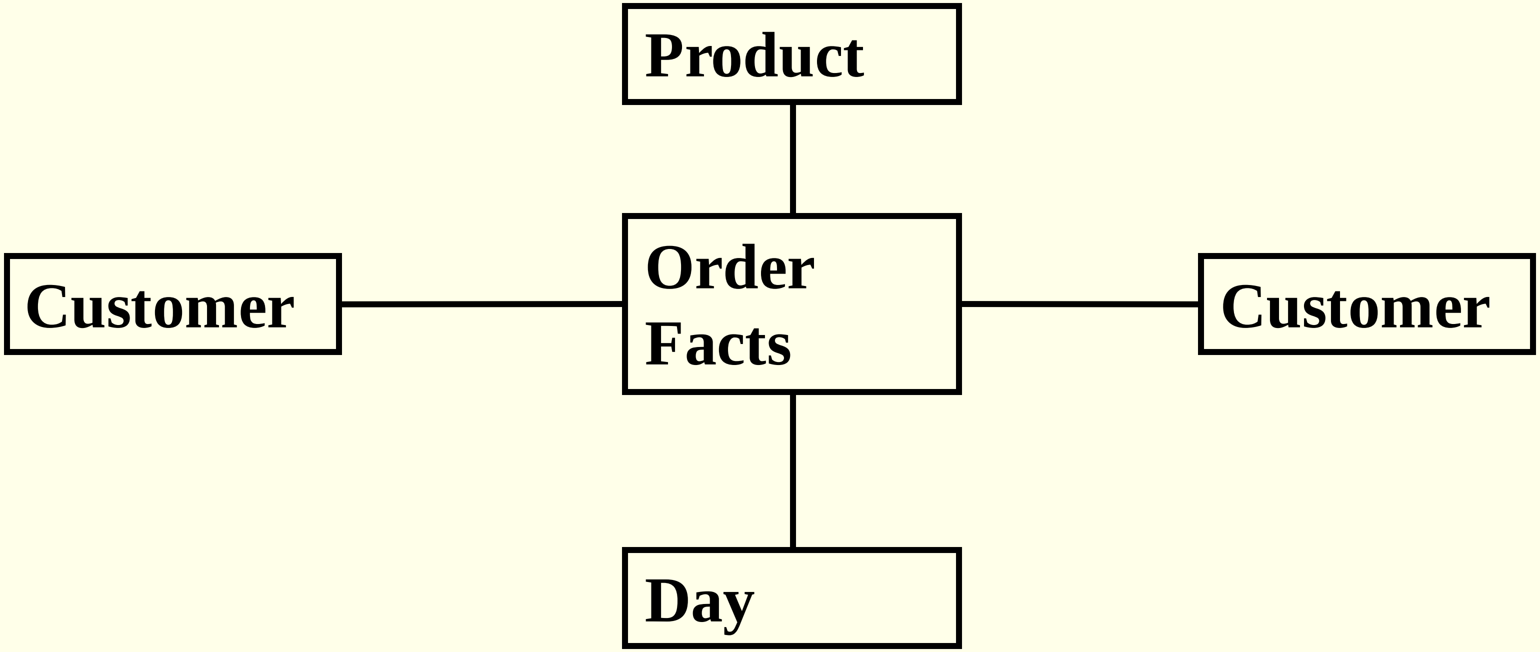


Grain: for each term we record each registration of a student in a course

Fact Tables - deep

Fact tables grow more quickly than dimensions.

Consider the schema below: order facts grow much faster than dimensions



Fact Tables - additivity

Measurements may be additive, semi-additive, non-additive

For some measurements care must be taken if we are going to add them across some dimension.

Sum(…) with Group By

Later … chapter 11 has more on this

Fact Tables – degenerate dimensions

If a dimension is stored in a fact table, the dimension is called a *degenerate* dimension

Transaction identifiers (order number, line number, registration number, …) often become degenerate dimensions.

Figure 3-5 … next slide

Fact Tables – degenerate dimensions

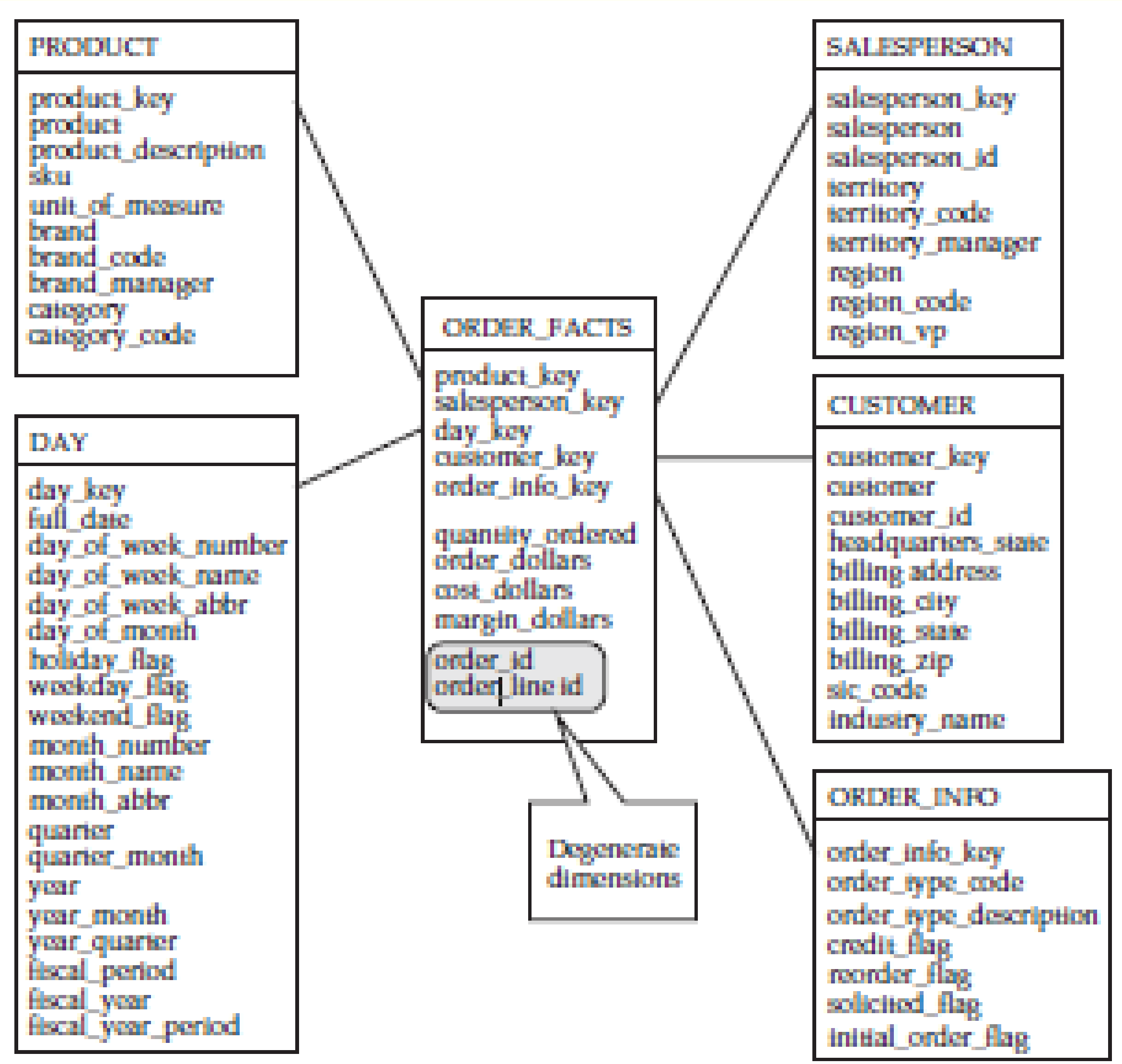


Figure 3-5 Degenerate dimensions define the grain of this fact table

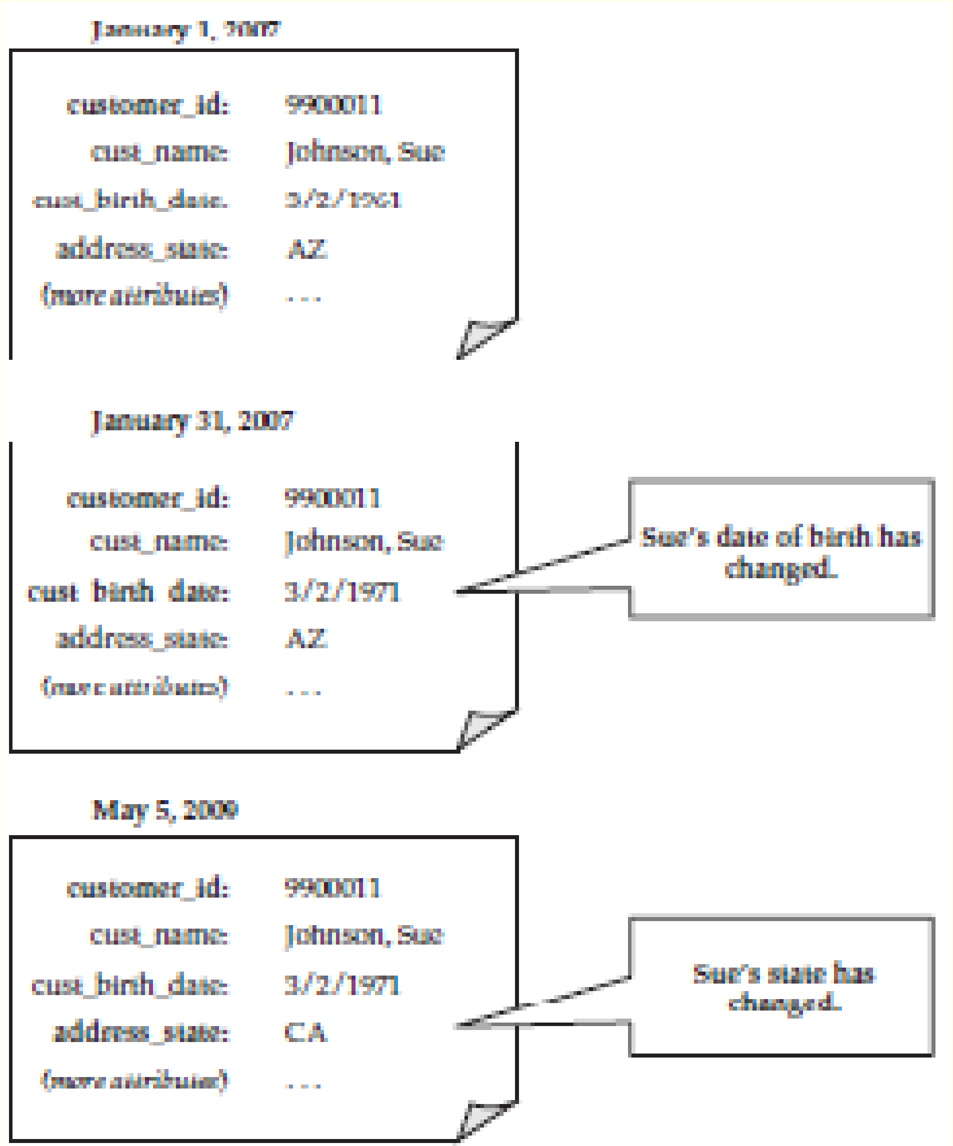
Assignment 1: includes orderId in the fact table

Slowly changing dimension techniques

- Data in source systems change.
- Changes must migrate to the warehouse.
- Each dimension needs a way to handle change.
- ETL must be designed appropriately

Slowly changing dimension techniques

Consider figure 3-6



Slowly changing dimension techniques

Type 1

When the source of a dimension value changes, and it is not necessary to preserve its history in the star schema, a type 1 response is employed.

Type 2

The type 2 change preserves the history of facts.

Facts that describe events before the change are associated with the old value; facts that describe events after the change are associated with the new value.

Slowly changing dimension techniques

Type 1

The dimension is simply overwritten with the new value. This technique is commonly employed in situations where a source data element is being changed to correct an error.

Type 2

When a type 2 change occurs, insert a new record into the dimension table. Any previously existing records are unchanged.

This type 2 response preserves context for facts that were associated with the old value, while allowing new facts to be associated with the new value.

A type 2 change results in multiple dimension rows for a given natural key.

More on Type 2 in chapter 8

Slowly changing dimension techniques

	Action	Effect on Facts
Type 1	Update Dimension	Restates History
Type 2	Insert New Row in Dimension Table	Preserves History

Figure 3-10 Summary of slowly changing dimension techniques

Slowly changing dimension techniques

ACS-4904 Winter 2020

Always include these 3 fields in a type 2 dimension:

Current indicator

Effective date

Expiry date

See fig 8-3 page 176

- These fields always have a value
- Effective/expiry dates establish non-overlapping date intervals specifying when a set of values were known/current.

Current indicator – *expired* / *current*

Expiry date – *current* row has the value *Dec 31, 9999*

Slowly changing dimension techniques

POLICY

policy_key	policy_number	policy_holder	transaction_type	effective_date	expiration_date	most_recent_version	marital_status	family_size	covered_parties
12882	40111	Smith, Hal	New Policy	2/14/2005	2/11/2006	Expired	Single	1	1
12911	40111	Smith, Hal	Policy Change	2/12/2006	3/30/2006	Expired	Married	2	1
13400	40111	Smith, Hal	Policy Renewal	3/31/2006	12/19/2007	Expired	Married	2	2
14779	40111	Smith, Hal	Policy Change	12/20/2007	2/3/2008	Expired	Married	3	3
14922	40111	Smith, Hal	Policy Change	2/4/2008	12/31/9999	Current	Married	4	4

Use to order a change history

Use for point-in-time analysis across policies

Use to filter for current status

```
SELECT
 policy_holder,
 transaction_type,
 marital_status
   :
ORDER_BY
 effective_date
```

```
SELECT
   policy_holder,
   marital_status
     :
WHERE
  12/31/2006 >= effective_date AND
  12/31/2006 <= expiration_date
```

```
SELECT
   policy_holder,
   marital_status
     :
WHERE
  most_recent_row
       = "Current"
```

Cubes

A dimensional model implemented as a

- Relational database …called a star schema
- Multidimensional database … called a cube
  - Aside: an article on Microsoft SQL Server Analysis Services: http://technet.microsoft.com/en-us/magazine/ee677579.aspx