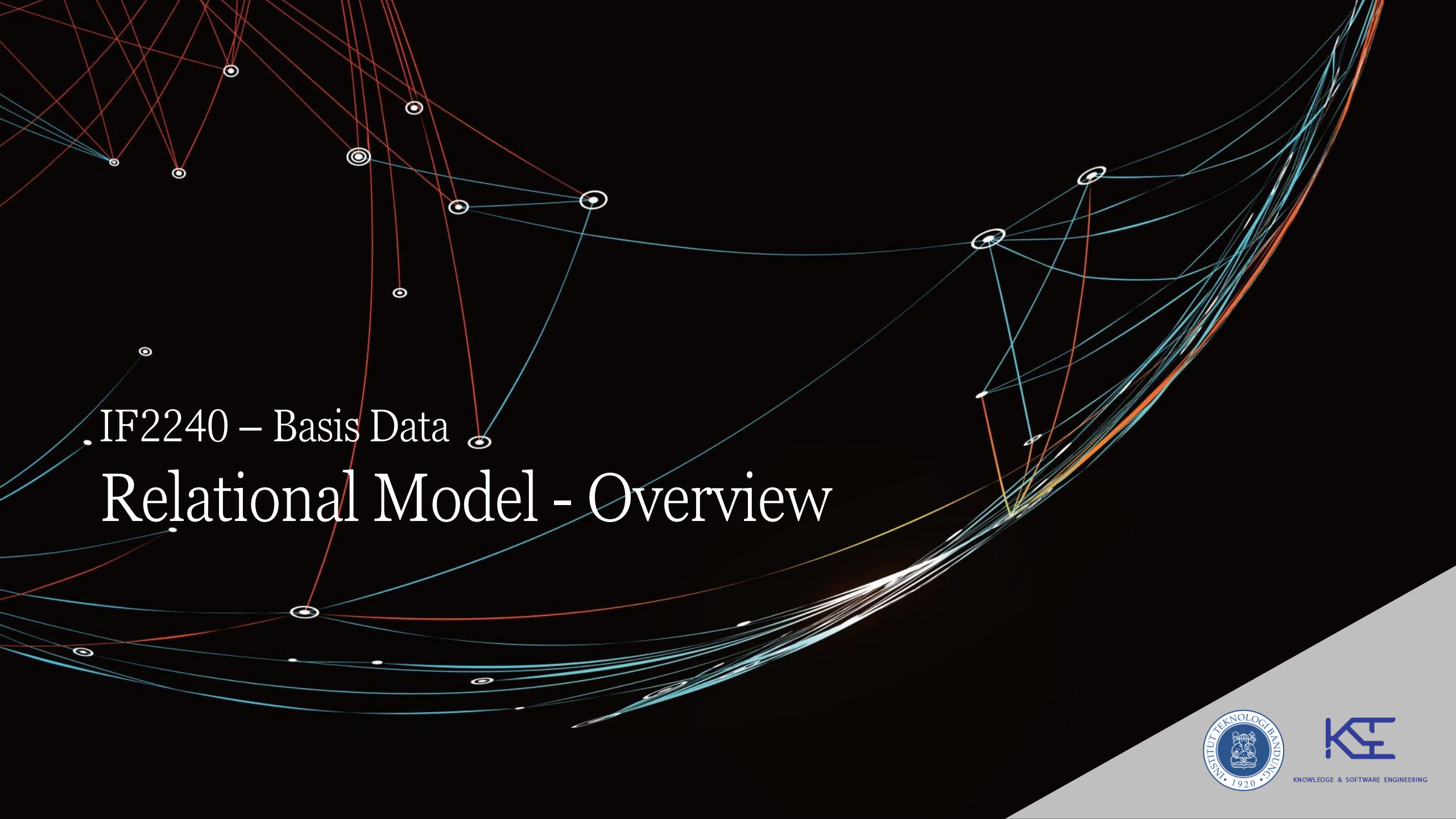
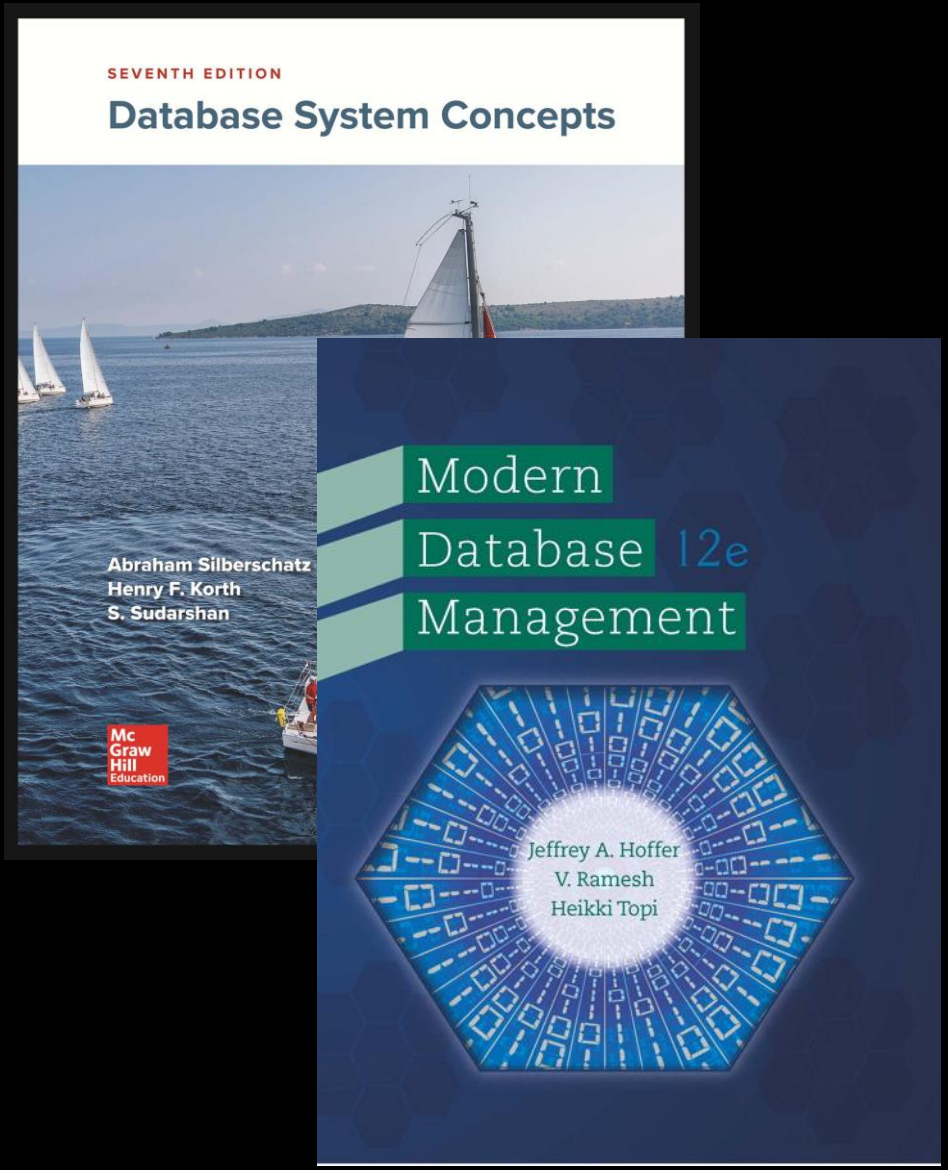


IF2240 – Basis Data

Relational Model - Overview

KNOWLEDGE & SOFTWARE ENGINEERING



References

Abraham Silberschatz, Henry F. Korth, S. Sudarshan : "**Database System Concepts**", 7th Edition

- Chapter 2: Introduction to the Relational Model

Jeffrey A. Hoffer, Mary B. Prescott, Heikki Topi : "**Modern Database Management**", 12th Edition

- Chapter 4: Logical Database Design and the Relational Model

Relational Model Concept

- A Relation is a mathematical concept based on the **ideas of sets**
- The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper:
  - "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970
- The above paper caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award
- Why study this? **Most widely used model**
  - Today, RDBMSs have become the dominant technology for database management, and there are literally hundreds of RDBMS products for computers ranging from smartphones and personal computers to mainframes

Relation

- Definition: A relation is a named, two-dimensional table of data
- Table consists of **rows (records)** and **columns (attribute or field)**
- Requirements for a table to qualify as a relation:
  - It must have a unique name
  - Every attribute value must be atomic (not multivalued, not composite)
  - Every row must be unique (can't have two rows with exactly the same values for all their fields)
  - Attributes (columns) in tables must have unique names
  - The order of the columns may be unordered
  - The order of the rows may be unordered

NOTE: all *relations* are in ***1st Normal form***

Example : Relation Instructor

attributes
(or columns)

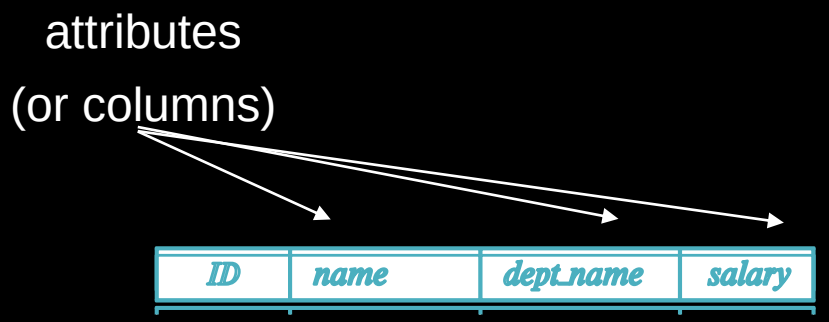
ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

KNOWLEDGE & SOFTWARE ENGINEERING

Attributes

- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
  - E.g. the value of an attribute can be an account number, but cannot be a set of account numbers
- The special value *null* is a member of every domain
- The null value causes complications in the definition of many operations
  - We shall ignore the effect of null values in our main presentation and consider their effect later
- Meanings for NULL values
  - Value unknown
  - Value exists but is not available
  - Attribute does not apply to this tuple (also known as value undefined)
- IMPORTANT: **NULL ≠ NULL**



Relation Schema & Instance

- A_1, A_2, …, A_n are *attributes*
- $R = (A_1, A_2, …, A_n)$ is a *relation schema*

  Example:

  instructor = (*ID*, *name*, *dept_name*, *salary*)

- The current values a relation are specified by a table
- An element ***t*** of relation ***r*** is called a *tuple* and is represented by a *row* in a table

Database Schema

Database schema -- is the logical structure of the database.

Database instance -- is a snapshot of the data in the database at a given instant in time.

Example:

- schema: *instructor* (*ID, name, dept_name, salary*)
- Instance:

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Why Split Information Across Relations?

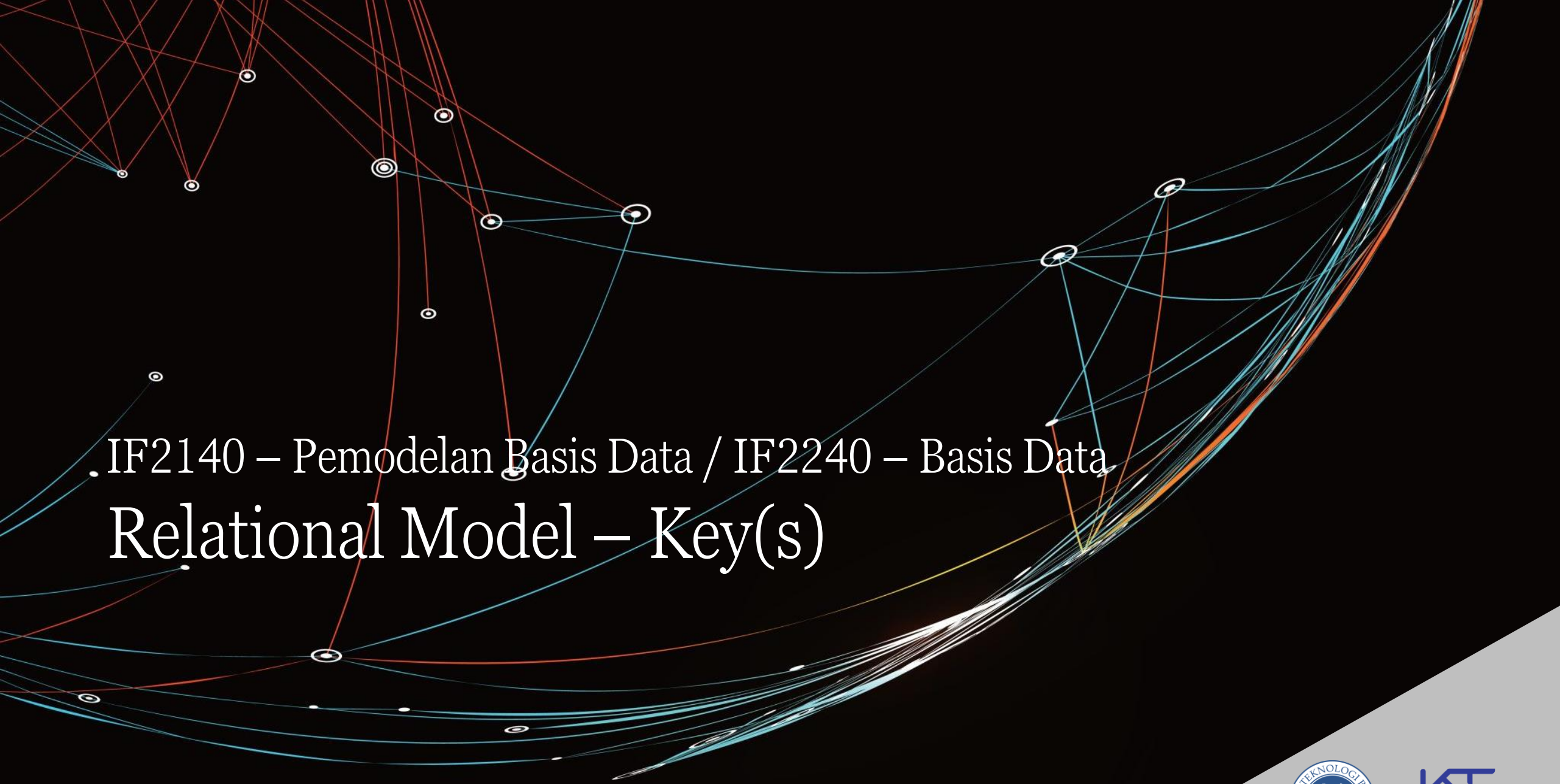
Storing all information as a single relation such as

bank(*account_number, balance, customer_name*, ..)

results in :

- repetition of information
  - e.g.,if two customers own an account (What gets repeated?)
- the need for null values
  - e.g., to represent a customer without an account

Normalization theory (we'll come back later) deals with how to design relational schemas



IF2140 – Pemodelan Basis Data / IF2240 – Basis Data

Relational Model – Key(s)

KNOWLEDGE & SOFTWARE ENGINEERING

Keys

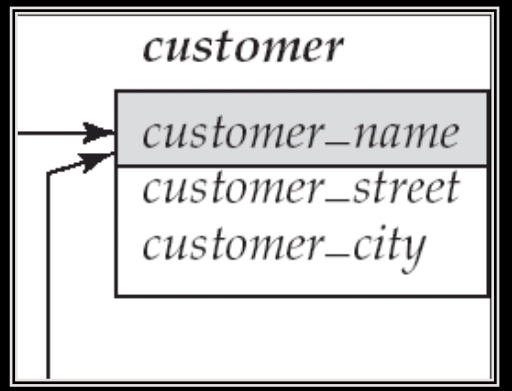
- Superkey
- Candidate Keys
- Primary Key
- Foreign Key

Superkeys

Let $K \subseteq R$

K is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*

- by "possible *r* " we mean a relation *r* that could exist in the enterprise we are modeling.
- Example: {*customer_name, customer_street*} and
  {*customer_name*}
  are both superkeys of *Customer*, if no two customers can possibly have the same name
  - In real life, an attribute such as *customer_id* would be used instead of *customer_name* to uniquely identify customers, but we omit it to keep our examples small, and instead assume customer names are unique.



Candidate and Primary Key

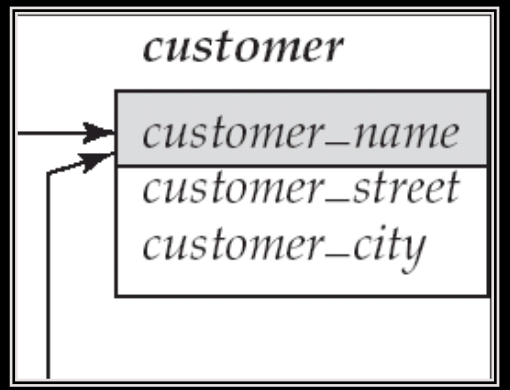
K is a **candidate key** if *K* is minimal
Example: {*customer_name*} is a candidate key for *Customer*, since it is a superkey and no subset of it is a superkey.

Primary key: a candidate key chosen as the principal means of identifying tuples within a relation

- Should choose an attribute whose value never, or very rarely, changes.
- E.g. email address is unique, but may change

Keys can be ***simple*** (a single field) or ***composite*** (more than one field)

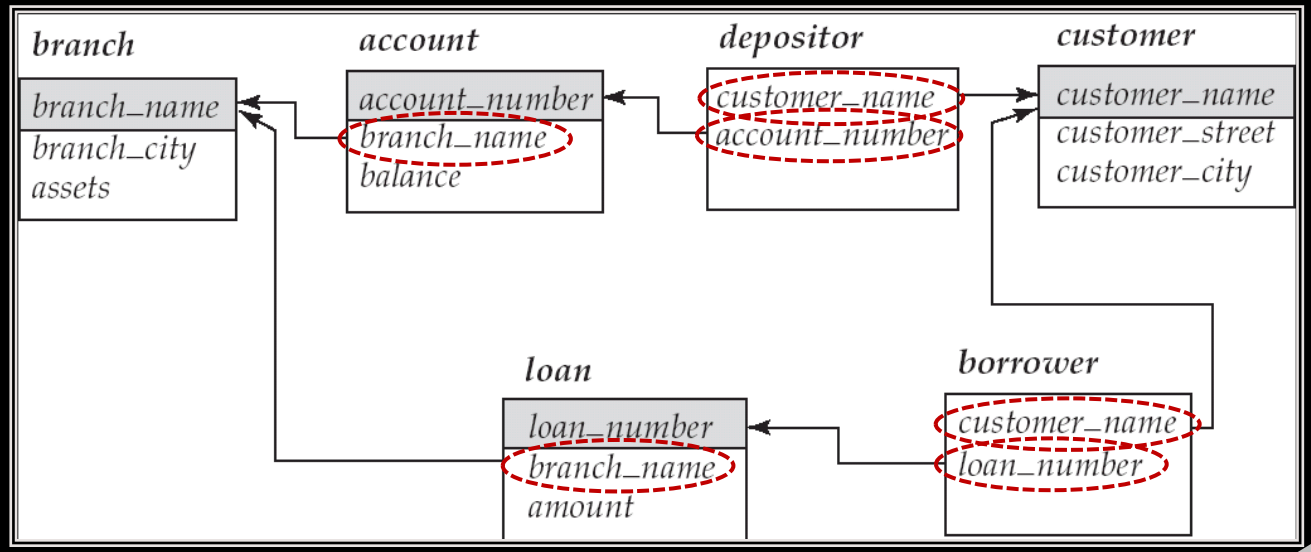
Keys usually are used as indexes to speed up the response to user queries



Foreign Keys

A relation schema may have an attribute that corresponds to the primary key of another relation. The attribute is called a **foreign key**.

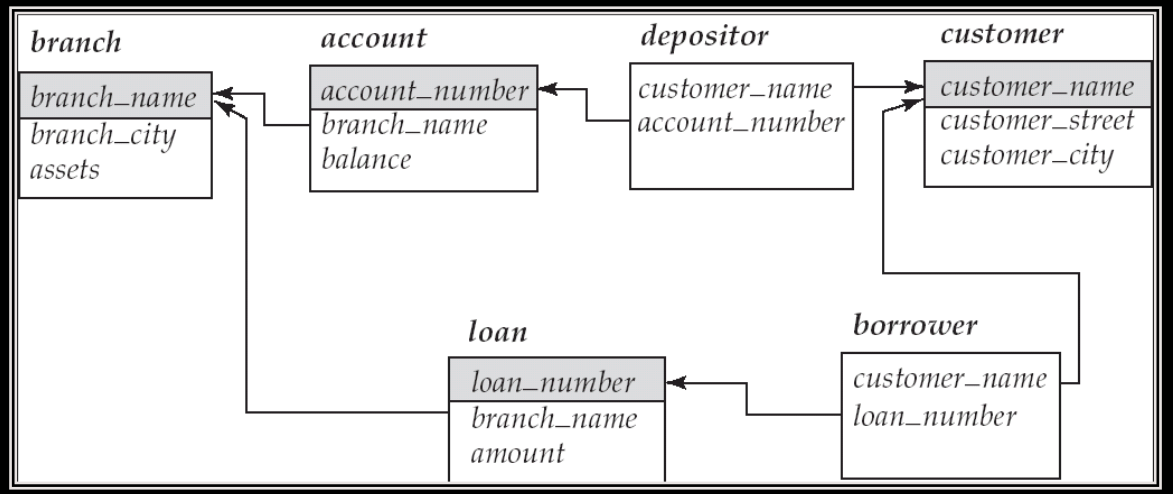
- E.g. *customer_name* and *account_number* attributes of *depositor* are foreign keys to *customer* and *account* respectively.
- Only values occurring in the primary key attribute of the **referenced relation** may occur in the foreign key attribute of the **referencing relation**.



Primary Keys & Foreign Keys (Example)

branch = (branch_name, branch_city, assets)
account = (account_number, branch_name, balance)
customer = (customer_name, customer_street, customer_city
loan = (loan_number, branch_name, amount)
depositor = (customer_name, account_number)
borrower = (customer_name, loan_number)

FK's: account(branch_name) → branch(branch_name)
loan(branch_name) → branch(branch_name)
depositor(customer_name) → depositor(customer_name)
depositor(account_number → account(account_number)
borrower(customer_name) → customer(customer_name)
borrower(loan_number) → loan(loan_number)



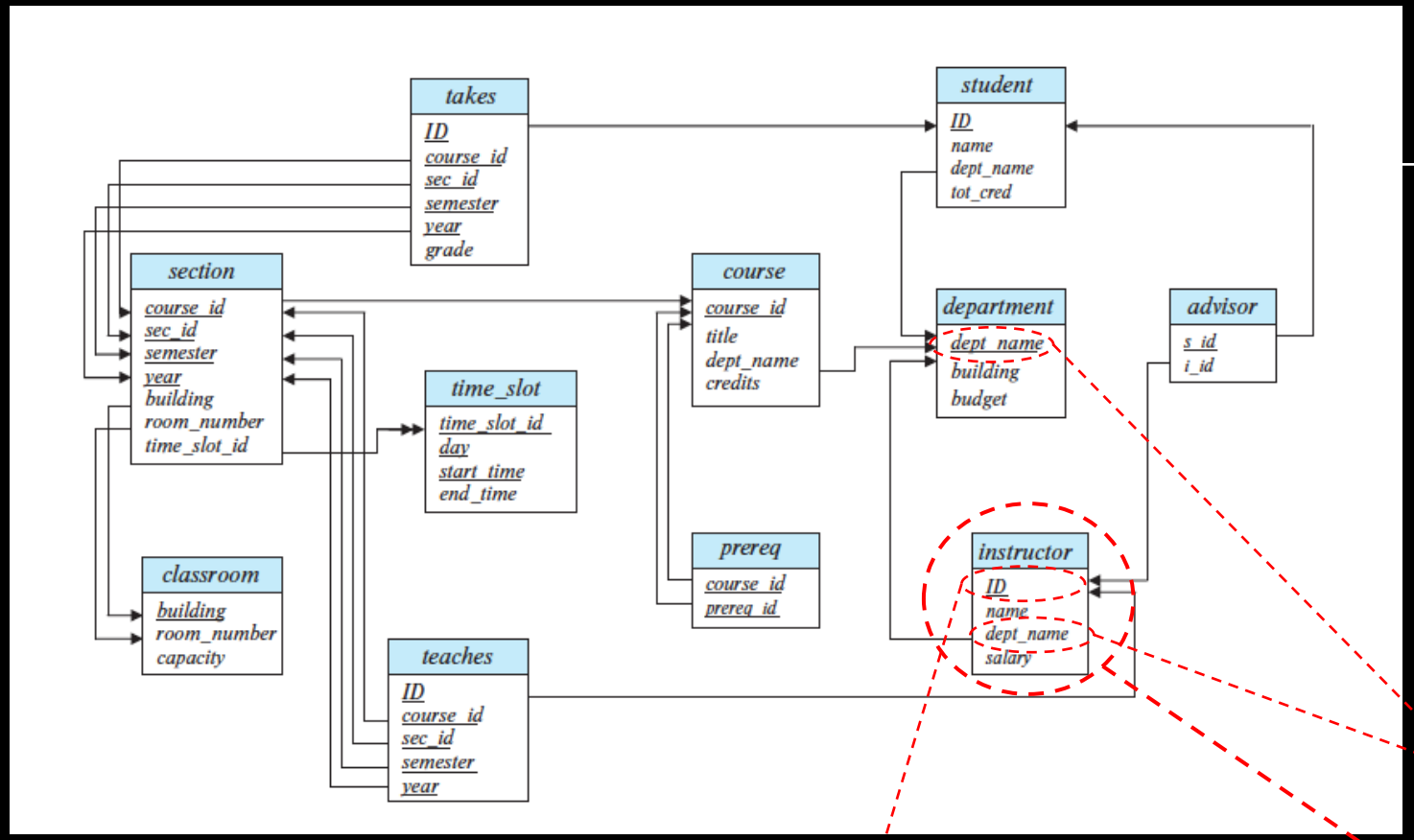
Note:
underlined : primary key

IF2140 – Pemodelan Basis Data / IF2240 – Basis Data

Relational Model – Schema Diagram & Integrity Constraint

KNOWLEDGE & SOFTWARE ENGINEERING

Schema Diagrams



Foreign-key constraints appear as arrows from the foreign-key attributes of the referencing relation to the primary key of the referenced relation

Primary-key shown underlined

Relation appears as box
Name at the top in the blue, attributes listed inside the box

Integrity Constraint

Domain Constraints

- All of the values that appear in a column of a relation must be from the same domain.

Entity Integrity

- The entity integrity rule is designed to ensure that every relation has a primary key and that the data values for that primary key are all valid.

Referential Integrity

- Rule that maintains consistency among the rows of two relations

Domain Constraint

A domain definition usually consists of the following components: domain name, meaning, data type, size (or length), and allowable values or allowable range (if applicable).

TABLE 4-1 Domain Definitions for INVOICE Attributes

Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
OrderID	Order IDs	Set of all possible order IDs	character: size 5
OrderDate	Order Dates	Set of all possible order dates	date: format mm/dd/yy
ProductID	Product IDs	Set of all possible product IDs	character: size 5
ProductDescription	Product Descriptions	Set of all possible product descriptions	character: size 25
ProductFinish	Product Finishes	Set of all possible product finishes	character: size 15
ProductStandardPrice	Unit Prices	Set of all possible unit prices	monetary: 6 digits
ProductLineID	Product Line IDs	Set of all possible product line IDs	integer: 3 digits
OrderedQuantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Entity Integrity

A rule that states that **no primary key attribute** (or component of a primary key attribute) may be null.

If we choose this one as a primary key, so there's no null CustomerID

TABLE 4-1 Domain Definitions for INVOICE Attributes

Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
OrderID	Order IDs	Set of all possible order IDs	character: size 5
OrderDate	Order Dates	Set of all possible order dates	date: format mm/dd/yy
ProductID	Product IDs	Set of all possible product IDs	character: size 5
ProductDescription	Product Descriptions	Set of all possible product descriptions	character: size 25
ProductFinish	Product Finishes	Set of all possible product finishes	character: size 15
ProductStandardPrice	Unit Prices	Set of all possible unit prices	monetary: 6 digits
ProductLineID	Product Line IDs	Set of all possible product line IDs	integer: 3 digits
OrderedQuantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Referential Integrity

Rule states that any foreign key value MUST match a primary key value in the referenced relation **(Or the foreign key can be null)**.

For example: Delete Rules

- **Restrict**–don't allow delete of "parent" side if related rows exist in "dependent" side
- **Cascade**–automatically delete "dependent" side rows that correspond with the "parent" side row to be deleted
- **Set-to-Null**–set the foreign key in the dependent side to null if deleting from the parent side (not allowed for weak entities)