The picture can't be displayed.

Relational Model

The original presentation is infused with more information and slides by Verena Kantere

Database System Concepts, 6th Ed.

Relational model

- **Relational schema**

EMPLOYEE

SSN	NAME	ADDRESS	SALARY

WORKS-ON

ESSN	PNUMB	HRSPW

PROJECT

NUMBER	NAME	LOCATION

- A set of relations, or tables with a name

- *The names of the columns are the names of the attributes*

Relational model

- Example database: Set of instances

EMPLOYEE

SSN	NAME	ADDRESS	SALARY
e1	john	Athens	300000
e3	mary	Patras	450000
e4	jack	Athens	145000

WORKS-ON

ESSN	PNUMB	HRSPW
e1	p1	10
e1	p2	15
e1	p5	30
e3	p4	20
e4	p4	40
e3	p2	25

PROJECT

NUMBER	NAME	LOCATION
p1	xyz	Rhodes
p2	rty	Athens
p4	hju	Patras
p5	ytu	Rhodes

Relational model: Actions

- **Actions/ Functions/ Operations**
  - They are distinguished as (a) Modifications, (b) Retrievals
- The set of actions in the Relational Model is closed, i.e. actions are defined on relations and produce relations
- ***MODIFY***
  - INSERT a tuple
  - DELETE a tuple
  - UPDATE a tuple
- The integrity constraints should be preserved with the execution of a modification action. That is why modifications may cause the execution of additional modifications.
  - E.g., when a tuple of EMPLOYEE is deleted, the tuples in WORKS_ON that have the same value in SSN are also deleted (non-existent employees should not work on projects)

Relational model: informal definition

- Proposed in 1970 by E.F. Codd (“*A relational model for large shared data banks*”*, CACM), as a Theory of Data Model*
- It was the motivation and the inspiration for many research efforts and ended up to be the most popular model.
- Today most DBMSs are relational (RDBMSs) and are available for all operating systems.

Relational model: informal definition

- A relational database is a set of relations
- ***Relation***:
  - *A table of values*
  - *Every column has a name and is called an **attribute***
  - *Every row is a called a **tuple** and represents the characteristics of an entity in the model*

- **From Codd's book: The Relational Model for Database Management:**

Given sets S1, S2, . . ., S*n* (not necessarily distinct), R is a relation on these *n* sets if it is a set of *n*-tuples, the first component of which is drawn from S1, the second component from S2, and so on.

More concisely, R is a subset of the Cartesian product S1 × S2 × . . . × S*n*. (For more information, see Chapter 4.) Relation R is said to be of *degree n*. Each of the sets S1, S2, . . ., S*n* on which one or more relations are defined is called a *domain*.

Relation example

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)

Relation example

Account

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Attribute Types

- 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
- The special value ***null*** is a member of every domain. Indicated that the value is "unknown"
- The null value causes complications in the definition of many operations

Relational model: formal definition*

* From the Database Systems Concepts Book

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

  Example:

  instructor = (*ID*, *name*, *dept_name*, *salary*)
- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation** ***r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

  Thus, a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$
- The current values (**relation instance**) of a relation are specified by a table
- An element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table

Relations are unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

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

Relation example - definition

customer-name = {Jones, Smith, Curry, Lindsay}
customer-street = {Main, North, Park}
customer-city = {Harrison, Rye, Pittsfield}

Then *r* = {

(Jones, Main, Harrison),
(Smith, North, Rye),
(Curry, North, Rye),
(Lindsay, Park, Pittsfield)

}

is a relation instance in *customer-name x customer-street x customer-city*

Relation

attributes
(or columns)

customer-name	customer-street	customer-city
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

tuples
(or rows)

customer

Characteristics of relations

- The order of attributes in a relation **is important**
- The order of tuples in a relation **is not important**
- Every tuple is stored **only once** in a relation (set)
- A value may appear **multiple times** in a column and is **atomic** - this is frequently referred to as **First Normal Form (1-NF)** relation
- The symbol for the value of *attribute* A_i *for a tuple* t is:

$$t[A_i] = v_i$$

Structural constraints

- Inherent constraints
  - Keys
    - The key is a property of the Relational Schema and not the Relation, which means that it holds for all instances of the schema, i.e. all possible such relations (Entity integrity)
  - Referential integrity – based on foreign keys
- Explicitly declared constraints
  - Domain constraints
  - Attribute constraints
  - User-defined constraints
  - Other explicit constraints, e.g. functional dependencies

Key constraints

- Let us assume $K \subseteq R$
- *K is a* **superkey** of the relational schema *R* if the values of *K* are enough to identify a unique tuple for every possible relation *r(R)*
  - E.g.: {*customer-name, customer-street*} and {*SSN*} are superkeys of *Customer*
- A **candidate key** K is a minimal superkey (or else key) (i.e. there is no subset of K that is a superkey). To K is also usually called a key.
  - E.g. SSN is a candidate key for Customer, but {SSN, NAME} is not
- A **primary key PK** is one of the candidate keys that is agreed that it will be the identifier for the tuples of a relation (primary keys are underlined)
  - E.g. <u>SSN</u> is the PK for EMPLOYEE.

Integrity Constraints

- The primary key PK in the relational schema R cannot have NULL values in the tuples of a relation r(R).

$$t[PK] \neq \text{NULL}, \ \text{for every} \ t \text{ in } r(R)$$

  - The reason is that the key is an identifier
  - Other attributes in R may be constrained so that they cannot have NULL values. This is done with explicit constraints.

Referential Integrity Constraints

- This constraint involves TWO relations and is used to ensure the consistency with a relationship set between tuples of the two relations.
- The most common form of the constraint is that of foreign keys.
  - A **foreign key** **FK** is a set of attributes in a relation schema R1 that is the primary key in another relation schema R2.

    A tuple t_1 in $r(R_1)$ is said that it refers to another tuple t_2 in $r(R_2)$, if:

$$t_1[FK] = t_2[FK]$$

e.g., EMPLOYEE (SSN, Name, BirthDate, Address, Sex, Salary, DNumber)

PROJECT (PNumber, PName, Location, DNumber)

WORKS_ON (SSN, PNumber, HoursPW)

Domain Constraints

- These are the rules that are defined by the domain and are inherited by the attributes that take values from the domain.
  - The domain can be defined together with integrity constraints (e.g. the domain of *integers* with all the constraints for integers). These are different **basic data types**.

Attribute Constraints

- These are additional constraints to the constraints of the domain and refer to the values of the attributes.
  - E.g., an attribute about ***small integers*** *or* ***integers between 1 and 10, etc***. *may have additional constraints to those of integers*

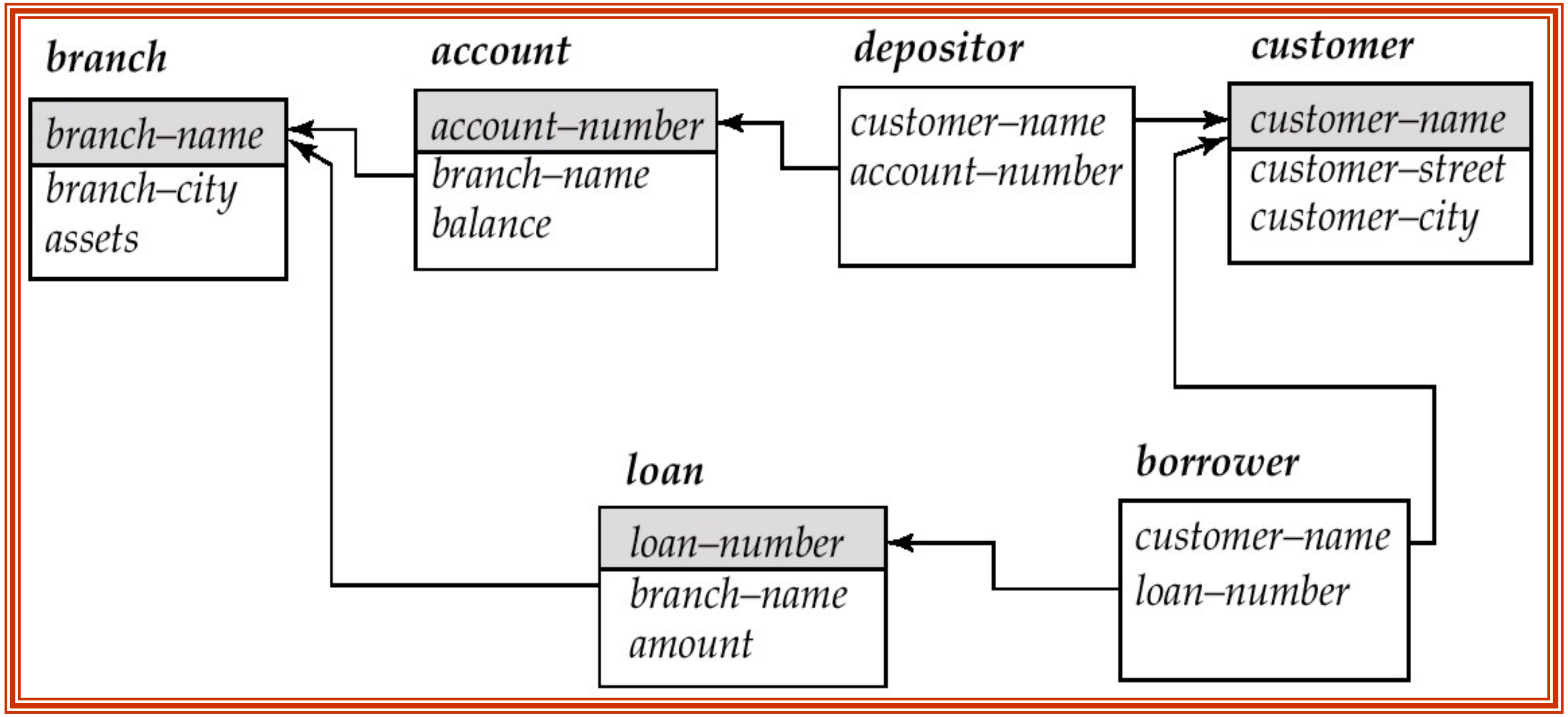
User-defined Constraints

- Every constraint beyond the ones mentioned before is called ***user-defined.***
- The support of business rules necessitates integrity constraints that have significant complexity
- These are defined either procedurally or declaratively
- A series of mechanisms are used in order to support such constraints in a relational system:
  - *stored procedures, triggers, methods (for object-oriented systems)*
- An important group of constraints is that of ***semantic integrity constraints***
- In general DBMSs are weak in the support of such constraints

Database for a Bank

branch (branch-name, branch-city, assets)
customer (customer-name, customer-street, customer-city)
account (account-number, branch-name, balance)
loan (loan-number, branch-name, amount)
depositor (customer-name, account-number)
borrower (customer-name, loan-number)

Relational model for a Bank



Branch Relation

branch-name	*branch-city*	*assets*
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

Borrower Relation

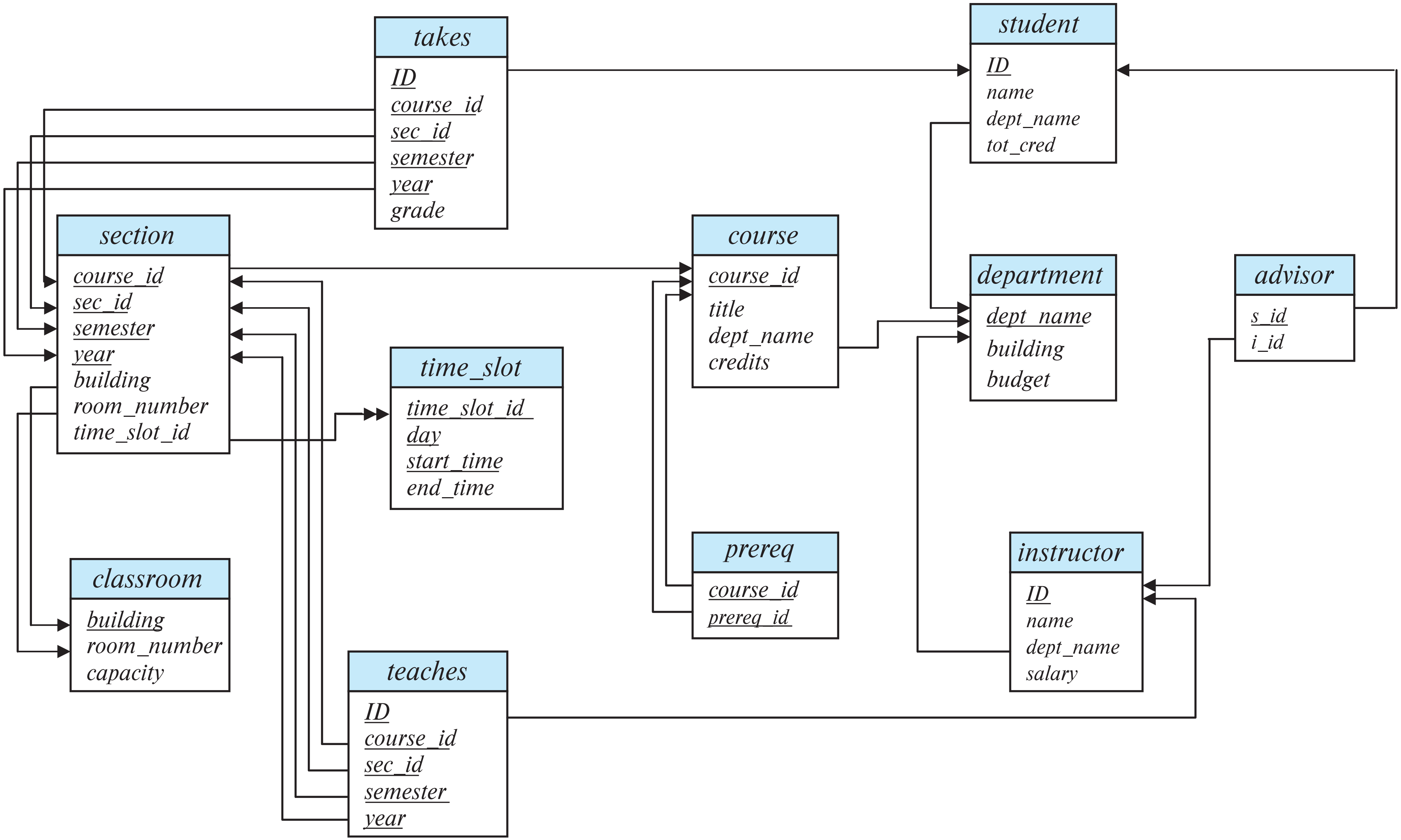
customer-name	*loan-number*
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Loan Relation

loan-number	*branch-name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Schema Diagram for University Database

Same figure from the 7th edition



The picture can't be displayed.

End of Chapter 2