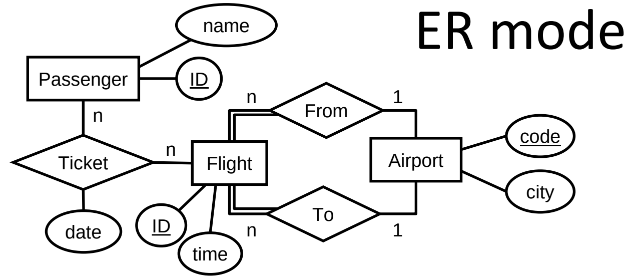
Contents

- Introduction and fundamentals
- Introduction to SQL
- Entity-relation model
- Relational model
- Relational design: normal forms
- Queries
  - Relational calculus
  - Relational algebra
- Database implementation
  - Physical structure: fields and records
  - Indexing
    - Simple indexes
    - B trees

ER model



Flight

number	origin	destination	time
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

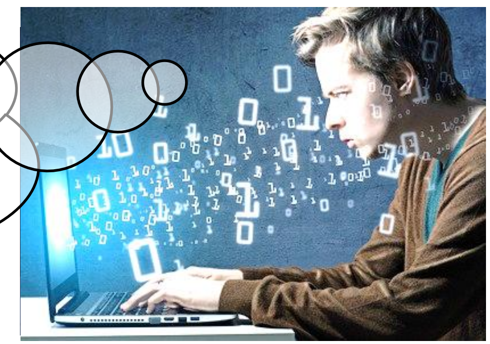
Passenger

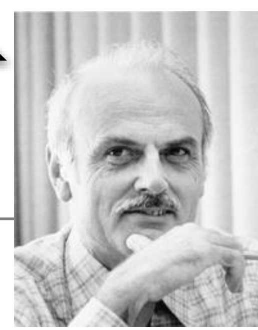
id	name
123	Mary
456	Peter
789	Isabel
321	Peter

Ticket

id	number	date	price
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

SQL

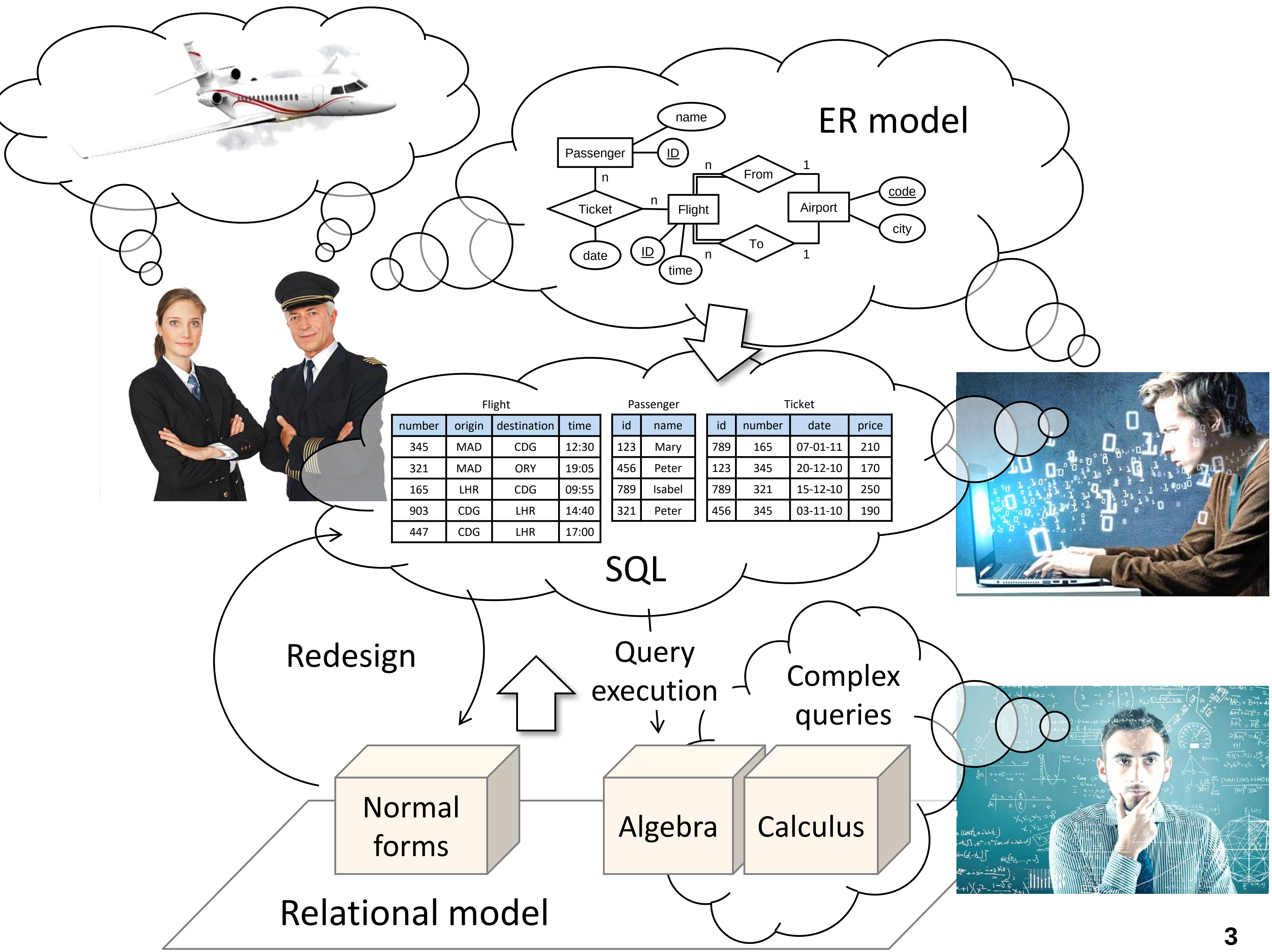




E. Codd

R. Boyce

Relational model



Flight

number	origin	destination	time
345	MAD	CDG	12:30
321	MAD	ORY	19:05
165	LHR	CDG	09:55
903	CDG	LHR	14:40
447	CDG	LHR	17:00

Passenger

id	name
123	Mary
456	Peter
789	Isabel
321	Peter

Ticket

id	number	date	price
789	165	07-01-11	210
123	345	20-12-10	170
789	321	15-12-10	250
456	345	03-11-10	190

Goal of this chapter

Understand that databases can be essentially described in set theory

Learn formalisms based on this model: normal forms, relational algebra, relational calculus

Relational model vs. SQL

- The relational model formalizes concepts implemented in SQL (or rather, SQL is an implementation of the relational model)
  - Schemas (table structure): attributes, domains
  - State of a schema (content of a table): tuples
  - Database, state of a database
  - Keys, superkeys, primary key, foreign key
- Notation: schema, tuple…
- Plus, upon the relational model, the following are formalized:
  - Normal forms: schema design properties
  - Queries: calculus and algebra

SQL (“tables”)

Table structure

```
create table Tweet (
  id : integer,
  content : text,
  author : integer
)
```

Table data

Tweet

id	content	author
7	‘Congrats!!…’	6
48	‘Notifications in…’	24
35	‘I just read the…’	6

Relational model

Relation schema

Attribute

Tweet (id, content, author)

Domain

Tweet (id : integer, content : string,
author : integer)

Attributes: atomic, univalued, unique name, admit NULL value

Relation state

r(Tweet) = { (7, ‘Congrats!!…’, 6),
(48, ‘Notifications in…’, 24),
(35,’I just read the…’, 6) }

r(Tweet) $\subset$ integer $\times$ string $\times$ integer

Set $\Rightarrow$ tuples are not repeated

r(Tweet) = { t1, t2, t3 }

t1 = (7, ‘Congrats!!…’, 6)   t1.id = 7 $\in$ integer

SQL ("tables")

Database = set of tables

Tweet

id	content	author
7	'Congratulations!!...'	6
48	'Notifications in...'	24
35	'I just read the first...'	6

User

id	name	email
24	Amelia	amy@gmail.com
6	James	jim@gmail.com
81	Nicholas	nick@gmail.com
73	Catherine	cate@gmail.com

Follows

user1	user2
6	24
73	6
81	73

...

Relational model

Database = set of relations + constraints

Database schema = set of relation schemas

Twitter = { Tweet, User, Follows... }

Database state = set of relation states

r(Twitter) = { r(Tweet), r(User), r(Follows)... }

SQL ("tables")	Relational model
Constraints	**Constraints**
Unique	(Candidate) key
Not NULL	Not NULL
Primary key	Primary key
References	Foreign key

- Superkey = set of attributes that contains a key
- Key = minimal superkey
- Primary key = key arbitrarily designated as such (only one per schema)
- Foreign key: integrity constraints
  - Points to a non-NULL key
  - The value exists in the referenced relation or the foreign key has value NULL
  - Integrity preservation

Summary of relational model: what do we need to know?

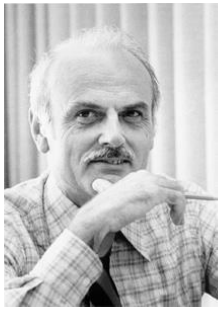
- Concepts
  - Schema, attribute, state, tuple, database
  - Emphasis: the state of a schema is a **set of** tuples
- Understand and handle the notation
- Conditions that attributes must satisfy
  - Unique name, values in the domain, atomic, univalued, admit NULL value
- Difference between keys, superkeys, primary key
  - Plus, primary keys cannot be NULL
- What does referential integrity mean with respect to foreign keys?
  - The referenced value must exist, or be NULL
- ER diagram conversion to relational schema

ER model vs. relational model

- Proposed by E. Codd in 1970
- ER $\cap$ MR
  - Entity / relationship $\rightarrow$ relation

    Entity type / relationship type $\rightarrow$ relational schema

    Entity extension / relationship extension $\rightarrow$ state of a relation
  - Attributes, domains
  - Superkeys, keys, primary key



Edgard F. Codd

- ER – MR
  - Multivalued attributes, composite attributes
  - Relation as a different element from entity
  - Weak entity (since there is no distinction between entity and relationship)
- MR – ER
  - Foreign keys
  - Database concept
  - Normalization, calculus, algebra
  - Directly expressible in SQL
- Some differences in notation, terminology, nuance
  - E.g. notion of constraint
  - Predicate logic nuance rather than set-based

Relational schema

- A relation name, and a list of attributes
  - Describes a relation
  - Similar to an entity in ER but predicate-based nuance rather than set-based (Cartesian product in E/R)
  - Relation arity: nr. of attributes
- Notation
  - R (A_1, A_2, …, A_n) where R is the relation name and A_k are the attributes
  - R (A_1 : dom (A_1), A_2 : dom (A_2), …, A_n : dom (A_n))
    where dom (A_k) is the domain of attribute A_k
- Example: User (username, email, name)

  User (username : string, email : string, name : string)

Relation attributes

- They have a name and an associated domain
  - Domains: string, numeric, postal code, etc.
  - Attributes can only take values in their domain
  - Two attributes of the same schema cannot have the same name
  - Attributes are understood to have a fixed place in the relation
- They can take the NULL value
  - It means the value does not exist, is not available, or is unknown
  - In general the less NULL values the better
- Equivalent to ER attributes but…
  - Atomic
  - Univalued

Keys

- Superkey
  - Set of attributes whose combination is unique for a relation
  - For instance, the set of all attributes of a relation schema is a (trivial) superkey
  - Examples: username + name is superkey of User
    Is pid a superkey of Person?
- Key
  - A minimal superkey, a.k.a. candidate key
  - Would be equivalent to UNIQUE in SQL
  - Examples: username + name is not a key for User
    username is a key
    email is a key
- Primary key
  - A key that is designated as primary for a relation schema
  - Is used in indexing (we will see this later on…)
  - Equivalent to PRIMARY KEY in SQL
  - The choice between candidate keys is arbitrary
  - Graphic notation: underlined

State of a relation

- $r\,(\mathrm{R}) \subset \mathrm{dom}\,(\mathrm{A}_1) \times \mathrm{dom}\,(\mathrm{A}_2) \times \cdots \times \mathrm{dom}\,(\mathrm{A}_n)$
- Set of tuples $r\,(\mathrm{R}) = \{t_1, t_2, \ldots, t_n\}$

  $t_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,n})$

  $x_{j,k} \in \mathrm{dom}(\mathrm{A}_k)$
- Notation

  $\mathrm{R}(x_1, \ldots, x_\mathrm{n})$ is the same as $(x_1, \ldots, x_\mathrm{n}) \in r\,(\mathrm{R})$

  $t\,[\mathrm{A}_k] = t\,[k] = t\,.\,\mathrm{A}_k = x_k$

  Subtuples: $t\,[\mathrm{A}_{k_1}, \ldots, \mathrm{A}_{k_j}] = t\,[k_1, \ldots, k_j] = (\mathrm{x}_{k_1}, \ldots, \mathrm{x}_{k_j})$

  where $k_i \in [1,n]$

  Also table notation (rows, columns and titles)

State of a relation

Relation Name

Attributes

STUDENT

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Database

- DB schema
  - Set of relational schemas $S = \{R_1, \ldots, R_m\}$
  - Set C of integrity constraints on schemas
- DB state
  - Set of states of each relation of the DB $\{r_1, \ldots, r_m\}$, where each r_k is a state of R_k
  - Where all r_k satisfy all the constraints in C
  - A state that does not meet all constraints is not valid
- We often refer to a DB as the schema plus its state

Constraints

- They apply to relation intension
  - Not enough that they be satisfied for just a specific relation state
- Inherent to the model (a.k.a. implicit)
  - E.g. no duplicate tuples are allowed
- Schema-specific (a.k.a. explicit)
- Data (a.k.a. functional) dependencies
  - They are the basis for normalization techniques (to be seen soon)
- Application-specific (a.k.a. semantic constraints or business rules)
  - Implemented in the application software, external to the DB

Schema constraints

- Domain constraints
  - Attributes are univalued
  - Their value must belong to the attribute domain
- Attribute constraints
  - **Keys**
    - Two tuples cannot take the same value on key attributes
    - Keys are minimal: if any of their attributes is removed the unicity is not mandatory
  - **NULL** value: it is possible to forbid the NULL value for specific attributes
- **Integrity** constraints
  - Of entities: no primary key can be NULL
  - Referential integrity…

Referential integrity

- Based on the foreign key notion
- They typically arise from relationships between entities in an ER model
- A set of attributes FK of a schema R_1 can be a **foreign key** that references R_2 if the attributes of FK have the same domains as the primary key of R_2
  - FK in R_1 is said to *reference* relation R_2
- A foreign key furthermore implies a **referential integrity** constraint:
  FK is a foreign key from R_1 to R_2 $\Rightarrow$ the values of FK in tuples of R_1 either occur in some tuple of R_2, o else they are NULL
- Referential **integrity preservation** in DB update operations
  - Insertion, deletion, modification
  - Reject, react (NULL, default, or propagate)

ER to relational model conversion

ER model	*Relational model*
Entity type E	Relational schema E
Attributes of E	Attributes of E, or schema apart
Atomic	Attribute of the relational schema E
Composite	An attribute of E for each atomic element
Multivalued	New relational schema with two attributes: primary key of the entity + attribute value
Weak entity E dependent on entities E_k	Schema where primary keys of E_k are added

ER to relational model conversion (cont)

ER model	*Relational model*
Relation R between E_1 and E_2	Relational schema R
Attributes of the relation R	Attributes of the relational schema R
R is *n*-*n*	Primary keys of E_1 and E_2 $\rightarrow$ attributes of R
R is *n*-1	Two options: a) Same as for *n*-*n* (especially if participation of E_1 is partial, to avoid NULLs) b) Add to the relational schema of E_1 the primary key of E_2 and the attributes of R (especially advised e.g. if relation is static)
R is 1-1	Two options: a) Same as for *n*-1 (especially if participation of E_1 or E_2 is partial) b) A single relational schema combining E_1 and E_2

Foreign keys

ER to relational model conversion (cont)

ER model	*Relational model*
Primary keys	
Entity	Same primary key
Weak entity	Partial key (if any) plus primary keys of identifying entities
Relation *n-n* betweenE_1 and E_2	Attributes of primary keys of E_1 and E_2 (and any attribute of R if needed)
Relation *n*-1 between E_1 and E_2	Primary key of E_1 (option b)
Relation 1-1 between E_1 and E_2	Primary key of E_1 or E_2 (option b)