An introduction to database design

Fabrice Rossi

CEREMADE
Université Paris Dauphine

2020

What is a good database?

What is a good database schema?

- some schemas are arguably bad
  - redundant schemas: repeated information is difficult to maintain
  - incomplete schemas: some information cannot be represented
- and can be (partially) fixed with specific algorithms (normalization)
- however
  - normalization cannot detect some instances of bad design
  - denormalization can be useful for performance reasons
  - completeness can only be considered with respect to design considerations

Example

Actors

id	first_name	last_name	gender	film_count
933	Lewis	Abernathy	M	1
2547	Andrew	Adamson	M	1
2700	William	Addy	M	1
2898	Seth (I)	Adkins	M	1
2925	Charles (I)	Adler	M	1

Directors

id	first_name	last_name
429	Andrew	Adamson
2931	Darren	Aronofsky
9247	Zach	Braff
11652	James (I)	Cameron
14927	Ron	Clements

Movies

id	name	year	rank
192017	Little Mermaid, The	1989	7.30
300229	Shrek	2001	8.10
306032	Snatch.	2000	7.90
333856	Titanic	1997	6.90

Directing

director_id	movie_id
429	300229
2931	254943
9247	124110
11652	10920
11652	333856

Genre

movie_id	genre
10920	Action
10920	Horror
10920	Sci-Fi
10920	Thriller
17173	Comedy

Roles

actor_id	movie_id	role
933	333856	Lewis Bodine
2547	300229	Duloc Mascot
2700	306032	Tyrone
2898	333856	Slovakian three-year-old boy
2925	192017	Additional Voices

Genres for directors

director_id	genre	prob
429	Adventure	0.75
429	Music	0.25
429	Fantasy	0.75
429	Romance	0.50
429	Family	0.75

Example

Problems

- redundant
  - film_count can be computed
  - prob can be computed
- incomplete/inconsistent
  - directors' gender?
  - can directors be actors?

Database design process

Overview

- domain expert interaction
  - data needs
  - functional requirements
- conceptual design
  - what are the entities described in the database?
  - how are they related one to another?
- logical design: translation of the conceptual design into a relational model
- physical design: storage and other aspects (out of the scope of this course)

Outline

Entity Relationship Model

Proposed by Peter Chen in 1976

Concepts

- Entity: uniquely identified object under study (e.g. a person)
- Relationship: a way to relate entities (e.g. a has access to b)
- Attribute: a property of an entity or of a relationship. An attribute has a domain (the set of values it can take)
- an ER model describes types, e.g. entity type (also called entity sets), not values

Example

Loan application data set

- `https://relational.fit.cvut.cz/dataset/Financial`
- 8 tables including
  - client table
  - account table
  - credit card table
  - loan table
  - etc.

Example

(part of the) Client table

client_id	gender	birth_date
1	F	1970-12-13
2	M	1945-02-04
3	F	1940-10-09
4	M	1956-12-01
5	F	1960-07-03

ER model

- entity type: client
- attributes
  - gender with domain F and M
  - birth_date with a date domain
  - client_id

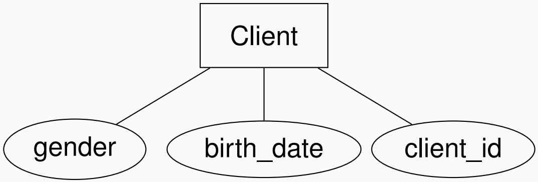
Example

(part of the) Client table

client_id	gender	birth_date
1	F	1970-12-13
2	M	1945-02-04
3	F	1940-10-09
4	M	1956-12-01
5	F	1960-07-03

ER model

- entity type: client
- attributes
  - gender with domain F and M
  - birth_date with a date domain
  - client_id



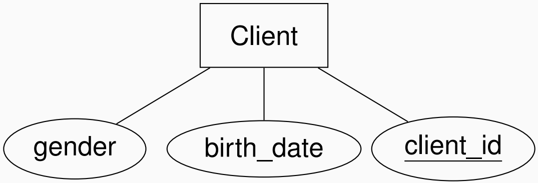
Example

(part of the) Client table

client_id	gender	birth_date
1	F	1970-12-13
2	M	1945-02-04
3	F	1940-10-09
4	M	1956-12-01
5	F	1960-07-03

ER model

- entity type: client
- attributes
  - gender with domain F and M
  - birth_date with a date domain
  - client_id key attribute



Graphical representation

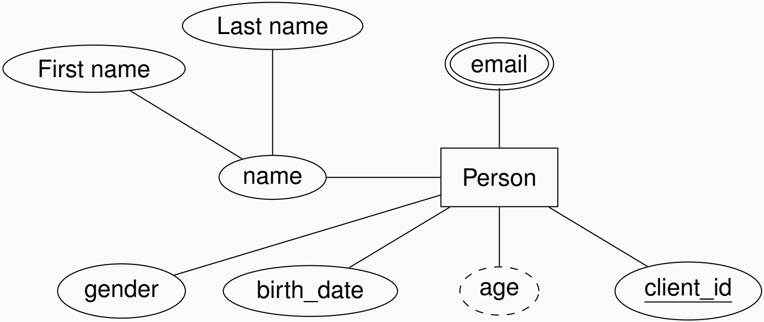
Entity type

Entity types are represented by rectangles link to attributes

Attribute type

- an ellipse per type
- key attribute type
  - a unique identifier of the corresponding entity
  - underlined in the representation
- composite attribute type
  - can be decomposed into sub-attributes
  - linked ellipses
- derived attribute type
  - can be computed from another one (e.g. age from birth date)
  - dashed border
- multi-valued attribute type
  - several values are authorized
  - double border

Example



Last name
First name
email
name
Person
gender
birth_date
age
client_id

Typical domains

Numerical

- integers
- decimal numbers
- possible constraints: positive numbers, number of significant digits, etc.

Temporal

- dates
- times

Textual

- words
- codes (such as post codes)
- structured strings (e.g. emails)

Others

- truth values (boolean)
- binary content (such as images)

Relationship

Principle

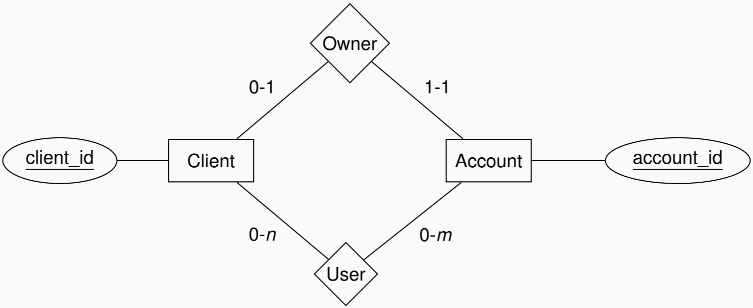
- a relationship represents an association between at least two entities (binary relationships are the most common)
- it can have attributes
- it is characterized by cardinalities
  - minimum and maximum number of relationships to which a given entity can participate
  - asymmetric
- a relationship type (also called relationship set) is graphically represented by a rhombus

Example

Loan application data set

- client entities and account entities
- relationships
  - a client can be the owner of an account
  - a client can be allowed to use an account
- cardinalities
  - owner:
    - each account has exactly one owner
    - each client can own at most one account (in this database)
  - user:
    - each account may have some users
    - each client can be the user of some accounts

Example



Outline

Conceptual design

Logical design

ER to relational

Mapping

- ER models are abstract
- must be mapped to a concrete database model
- the relational model is close enough to ER to enable a simple mapping strategy

Principles

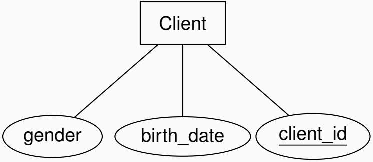
- Entity type $\rightarrow$ relation schema
- Simple attribute type $\rightarrow$ relational attribute
- Key attribute type $\rightarrow$ primary or alternative key
- All the rest (relationship types and complex attribute types) $\rightarrow$ relation schemas and keys

Mapping Entity Types

With simple attribute types

- direct mapping
- an entity type is mapped to a relation schema
- each attribute type corresponds to a relational attribute
- a key is mapped to a primary or alternative key
- composite attribute types are mapped to a set of relational attributes

ER model



Relation schema

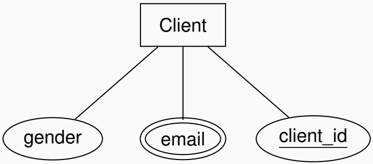
Client(gender, birth_date, **client_id**)

Multi-valued attribute types

Method

- a multi-valued attribute type cannot be mapped to a column type because of the domain integrity constraint
- representation via a relation schema
  - a relation schema per multi-valued attribute type
  - a relational attribute for the attribute
  - a foreign key to map back the attribute to the entity

ER model



Relation schemas

- Client(gender, **client_id**)
- ClientEmail(**email**, client_id)

Mapping relationship types

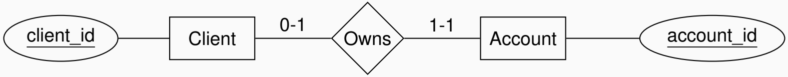
Principles

- the mapping depends on the cardinalities of the relationship type
- 0-1 cardinalities on a least one side foreign key
- other cases: foreign keys or specific relational schema

One to one (1:1) relationship type

- each side is 0-1 or 1-1
- mapped to a foreign key:
  - in the 1-1 relation type if it exists with a non *nullable* relational attribute
  - *nullable* if both side are 0-1

Example



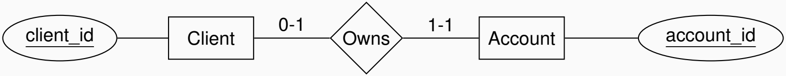
Relation schemas

- Client(**client_id**)
- Account(**account_id**,client_id)
- client_id is non *nullable* in Account

Inferior alternative

- Client(**client_id**,account_id)
- account_id is *nullable* in Client
- Account(**account_id**)

Example



Relation schemas

- Client(**client_id**)
- Account(**account_id**,client_id)
- client_id is non *nullable* in Account

Cardinalities

- Owns $\leftrightarrow$ Account is enforced
- but an account can be shared by several clients
- Client $\leftrightarrow$ Owns is $0 - m$

Inferior alternative

- Client(**client_id**,account_id)
- account_id is *nullable* in Client
- Account(**account_id**)

Cardinalities

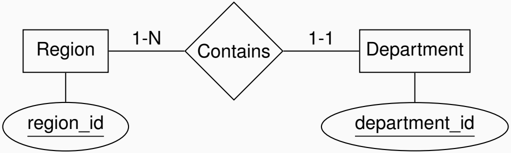
- Client $\leftrightarrow$ Owns is enforced
- but an account can be assigned to any number of clients
- Owns $\leftrightarrow$ Account is $0 - m$

Higher cardinalities

One to *N* (1:*N*) relationship type

- mapped to a foreign key in the 1 side entity relation schema
- *nullable* if this side is 0 – 1, non nullable if it is 1 – 1
- the minimal cardinality on the *N* side cannot be enforced

Example



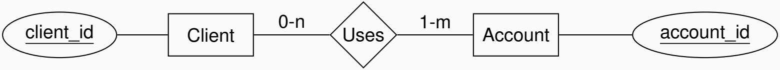
- Department(**departement_id**,region_id)
- Region(**region_id**)

Higher cardinalities

M to *N* (*M*:*N*) relationship type

- mapped to a relation type
- two foreign keys, one for each table
- the primary key of the relation type is the combination of the foreign keys
- minimal cardinalities cannot be enforced

Example



- Client(**client_id**)
- Account(**account_id**)
- Uses(**client_id, account_id**) both non nullable

Normal forms

Principle

- normal forms are “good design” constraints
  - a database (schema) is in a given normal form if it fulfills the corresponding constraints
  - basic normal forms are core aspects of the relational model
  - advanced normal forms enforce good design (no redundancy)
- normalization
  - process that turns an unnormalized database into a normalized one
  - generally works by *decomposition*: split a relation into multiple relations
  - leverages knowledge about the data (especially when operating as the schema level)

First normal form

1NF

- proposed by E. Codd in 1971
- a relation schema is in first normal form if all its attributes have **atomic domains**
- normalization by splitting

Atomic domain

- a domain is atomic if its elements are indivisible units
- non atomic domains
  - sets or lists of values: several emails or phone number in a *single* attribute (e.g. multi-valued attributes)
  - structured objects such addresses (city, street name, etc.)
  - french social security number
- somewhat ambiguous notion: is `toto@domain.com` atomic?

Examples

Structured domain

- split the domain into several domains
- e.g. address into
  - country
  - city
  - zipcode
  - street name
  - number
  - etc.

Multi-valued attribute

- use one tuple per value
- **do not** use one attribute per value

Original

name	email
Toto	Toto@d1.com, Toto@d2.com

Examples

Structured domain

- split the domain into several domains
- e.g. address into
  - country
  - city
  - zipcode
  - street name
  - number
  - etc.

Multi-valued attribute

- use one tuple per value
- **do not** use one attribute per value

Original

name	email
Toto	Toto@d1.com, Toto@d2.com

Do not do that!

name	email1	email2
Toto	Toto@d1.com	Toto@d2.com

Examples

Structured domain

- split the domain into several domains
- e.g. address into
  - country
  - city
  - zipcode
  - street name
  - number
  - etc.

Multi-valued attribute

- use one tuple per value
- **do not** use one attribute per value

Original

name	email
Toto	Toto@d1.com, Toto@d2.com

1NF

name	email
Toto	Toto@d1.com
Toto	Toto@d2.com

Functional dependencies

Specifying constraints

- functional dependencies represent constraints associated to the context
- definition
  - $R(\mathcal{A})$ a relation schema with attributes $\mathcal{A} = \{A_1, \ldots, A_K\}$
  - $\alpha \subset \mathcal{A}$ and $\beta \subset \mathcal{A}$
  - a **functional dependency** is denoted $\alpha \rightarrow \beta$
  - $\alpha \rightarrow \beta$ holds on r an instance of $R(\mathcal{A})$ if

$$\forall t_1 \in r, t_2 \in r, t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

  - in informal terms, the attributes α uniquely determine the attributes β
- property: if K is a super key of r an instance of $R(\mathcal{A})$ then $k \rightarrow \mathcal{A}$ holds on r

Example

French population INSEE file (single relation)

Code région	Nom de la région	Code département	Code arrondissement	Code canton	Code commune
76	Occitanie	46	2	16	195
76	Occitanie	34	2	09	010
76	Occitanie	34	3	23	248
32	Hauts-de-France	62	1	09	671
76	Occitanie	46	3	04	308

Nom de la commune	Population municipale	Population comptée à part	Population totale
Missy-lès-Pierrepont	110	6	116
La Villeneuve-en-Chevrie	601	8	609
Lawarde-Mauger-l'Hortoy	181	0	181
Valleroy-le-Sec	169	5	174
Vitray	101	2	103

Some functional dependencies

- (Code département, Code Commune) is the primary key
- Code région $\rightarrow$ Nom de la région
- Code département $\rightarrow$ (Code région, Nom de la région)
- (Population municipale, Population comptée à part) $\rightarrow$ Population totale

Second normal form

2NF

- a relation r instance of $R(\mathcal{A})$ is in the second normal form if
  1. $R(\mathcal{A})$ is in the first normal form and
  2. if a functional dependencies $\alpha \rightarrow \beta$ holds on r then
     - α is not a strict subset of the primary key of $R(\mathcal{A})$
     - or β contains only attributes that belong to a candidate key of r
- in informal terms, when an attribute A_k is not part of a candidate key of r, then it depends on the full key, not a subset of the key
- design problem
  - redundancy
  - A_k is not unique: values are repeated
  - if $\alpha \rightarrow A_k$ and α is only a subset of the key, α is not unique!
  - we should gather (α, A_k) in another relation

Example

French population INSEE file (single relation)

Code région	Nom de la région	Code département	Code arrondissement	Code canton	Code commune
11	Île-de-France	93	2	14	051
11	Île-de-France	93	1	06	010
11	Île-de-France	93	1	15	055
11	Île-de-France	93	2	19	078
11	Île-de-France	93	2	20	074

Nom de la commune	Population municipale	Population comptée à part	Population totale
Noisy-le-Grand	64619	521	65140
Bondy	53074	307	53381
Pantin	54852	323	55175
Villepinte	35864	198	36062
Vaujours	6867	167	7034

2NF constraint not enforced

- (Code département, Code Commune) is the primary key
- Code région is not in a candidate key
- but Code département $\rightarrow$ (Code région, Nom de la région)

Decomposition

Principle

- starting schema $R(\mathcal{A})$
- two attribute subsets $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$
- a decomposition of r instance of $R(\mathcal{A})$ over $\mathcal{A}_1$ and $\mathcal{A}_2$ is the pair of relations

$$r_1 = \Pi_{\mathcal{A}_1}(r) \qquad\qquad r_2 = \Pi_{\mathcal{A}_2}(r)$$

- a decomposition is **lossless** if

$$r = r_1 \bowtie r_2 = \Pi_{\mathcal{A}_1}(r) \bowtie \Pi_{\mathcal{A}_2}(r)$$

- notice that is $\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$, the natural join is the cartesian product

Decomposition and normalization

Principle

- if $r \in R(\mathcal{A})$ is decomposed in a lossless way over $\mathcal{A}_1$ and $\mathcal{A}_2$ then either
  - $\mathcal{A}_1 \cap \mathcal{A}_2 \rightarrow \mathcal{A}_1$
  - or $\mathcal{A}_1 \cap \mathcal{A}_2 \rightarrow \mathcal{A}_2$
- in the relational model if e.g. $\mathcal{A}_1 \cap \mathcal{A}_2 \rightarrow \mathcal{A}_1$
  - $\mathcal{A}_1 \cap \mathcal{A}_2$ is the primary key of $\Pi_{\mathcal{A}_1}(r)$
  - $\mathcal{A}_1 \cap \mathcal{A}_2$ is a foreign key from $\Pi_{\mathcal{A}_2}(r)$ to $\Pi_{\mathcal{A}_1}(r)$
- can be leveraged to enforce normal form(s)

Example

Possible solution with two relations

Code département	Code arrondissement	Code canton	Code commune
93	2	14	051
93	1	06	010
93	1	15	055
93	2	19	078
93	2	20	074

Nom de la commune	Population municipale	Population comptée à part	Population totale
Noisy-le-Grand	64619	521	65140
Bondy	53074	307	53381
Pantin	54852	323	55175
Villepinte	35864	198	36062
Vaujours	6867	167	7034

and

Code département	Code région	Nom de la région
01	84	Auvergne-Rhône-Alpes
02	32	Hauts-de-France
03	84	Auvergne-Rhône-Alpes
04	93	Provence-Alpes-Côte d'Azur
05	93	Provence-Alpes-Côte d'Azur

Trivial depencies and closure

Definitions

- if $\beta \subset \alpha$ then $\alpha \to \beta$ and the dependency is *trivial*
- if *F* is a set of functional dependencies, F^+ is its *closure* defined as the smallest set of functional dependencies *S* such that
  - $F \subset S$
  - if $\alpha \to \beta \in S$ and $\beta \to \gamma \in S$, then $\alpha \to \gamma \in S$
  - for all α, $\beta \subset \alpha$, $\alpha \to \beta \in S$
  - if $\alpha \to \beta \in S$, for all δ, $\alpha\delta \to \beta\delta \in S$

Example

- if $F = \{$ Code département $\to$ Code région, Code région $\to$ Nom de la région $\}$
- then F^+ contains Code département $\to$ Nom de la région and many others

Third normal form

3NF

- a relation *r* instance of $R(\mathcal{A})$ is in the third normal form if
  1. *r* is in the second normal form with respect to *F* and
  2. for all $\alpha \rightarrow \beta \in F^+$, at least one of the following property is true
     - $\alpha \rightarrow \beta$ is trivial
     - α is a super key of *r*
     - each attribute *A* in $\beta - \alpha$ is contained in a candidate key of *r*
- design problem
  - redundancy
  - not covered by the 2NF (more general dependencies)
  - non trivial $\alpha \rightarrow \beta$ when α is not a super key: α is repeated and so is β
  - in addition β is not unique even combined by other attributes

Example

Main relation from the 2NF city database

Code département	Code arrondissement	Code canton	Code commune
93	2	14	051
93	1	06	010
93	1	15	055
93	2	19	078
93	2	20	074

Nom de la commune	Population municipale	Population comptée à part	Population totale
Noisy-le-Grand	64619	521	65140
Bondy	53074	307	53381
Pantin	54852	323	55175
Villepinte	35864	198	36062
Vaujours	6867	167	7034

3NF constraint not enforced

- (Population municipale, Population comptée à part) $\rightarrow$ Population totale
- (Population municipale, Population comptée à part) is not a super key
- Population totale is not part of a super key

Example

Theoretical solution

- remove Population totale from the main relation
- create relation with (Population municipale, Population comptée à part, Population totale) using (Population municipale, Population comptée à part) as the primary key

In practice

- Population totale=Population municipale + Population comptée à part
- remove Population totale from the relation!

Boyd-Codd Normal Form

BCNF

- a relation *r* instance of $R(\mathcal{A})$ is in the Boyd-Codd normal form if
    1. *r* is in the second normal form with respect to *F* and
    2. for all $\alpha \rightarrow \beta \in F^+$, at least one of the following property is true
        - $\alpha \rightarrow \beta$ is trivial
        - α is a super key of *r*
- 3NF with additional restriction
    - trade off between redundancy and *dependency preservation*
    - a database can always be put in 3NF with dependency preservation (i.e. functional dependencies can be verified relation by relation)
    - a database can always be put in BCNF but not always with dependency preservation

Changelog

- November 2020: initial version

Licence

Version

Last git commit: 2020-12-08
By: Fabrice Rossi (Fabrice.Rossi@apiacoa.org)
Git hash: f4c571dde251990da4b13badf5b505a8ef2647f6