Conceptual Modeling, Mapping to Relational, Schema validation

Database and Web Applications Laboratory

João Correia Lopes

INESC TEC, Faculdade de Engenharia, Universidade do Porto

6 March 2020

GOALS

By the end of this class, the student should be able to:

- Obtain the Conceptual Data Model
- Obtain the Relational Schema from the Conceptual Model
- Validate the Relation Schema using normalisation

PREVIOUSLY IN LBAW

- ER :: Requirements Specification and User Interfaces
  - A1: Project presentation
  - A2: Actors and User stories
  - A3: User Interfaces Prototype

⇒ MediaLibrary Requirements

NOW IN LBAW

- EBD :: Database specification
  - A4: Conceptual Data Model
  - A5: Relational Schema, validation

⇒ `http://web.fe.up.pt/~jlopes/doku.php/teach/lbaw/artefacts/`

Part I

Conceptual Modeling (A4)

CONTENTS

⇒ `http://web.fe.up.pt/~jlopes/doku.php/teach/lbaw/artefacts/a04`

Conceptual Data Model (A4)

A4

This artefact contains the identification and description of the entities of the domain and the relationships between them.

- The **data requirements** of the system are detailed
- The Conceptual Domain Model is simplified to include only concepts of the domain that are stored in the database
- The Conceptual Model is obtained by using a UML class diagram containing the classes, associations, cardinality and roles
- For each class, the attributes, associations and constraints are included in the class diagram
- Business rules not included in the class diagram are included as UML notes

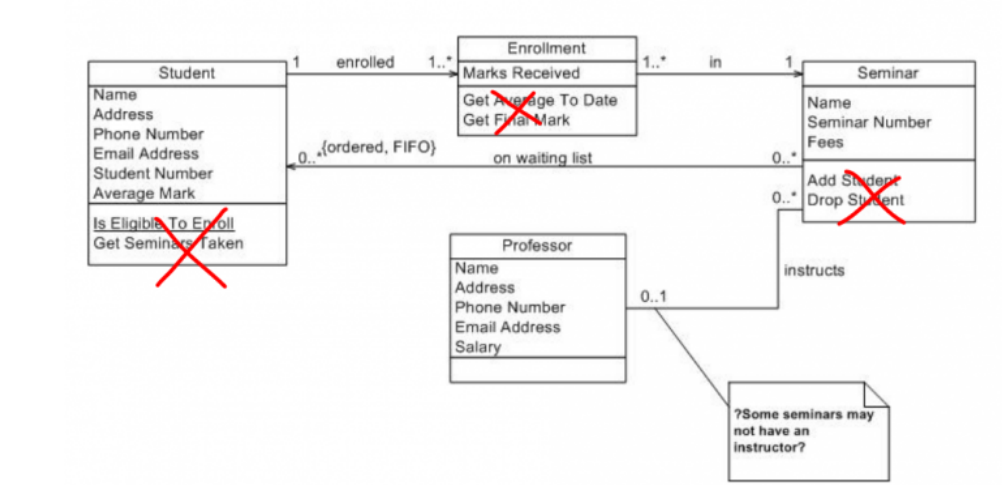
Conceptual Data Model (A4)

A4

This artefact contains the identification and description of the entities of the domain and the relationships between them.

- The **data requirements** of the system are detailed
- The Conceptual Domain Model is simplified to include only concepts of the domain that are stored in the database
- The Conceptual Model is obtained by using a UML class diagram containing the classes, associations, cardinality and roles
- For each class, the attributes, associations and constraints are included in the class diagram
- Business rules not included in the class diagram are included as UML notes

UML CLASS DIAGRAM



Scott Ambler. The Object Primer. Cambridge University Press, 3rd Edition, 2004. Fig. 8.10

LOGICAL DATA MODELS (LDM)

- Data modeling is the act of exploring data-oriented structures
- Data models are used for a variety of purposes, from conceptual models to physical design models
- Logical data models are used to explore **domain concepts and their relationships**
- With data modeling, you identify **data entities** and assign data attributes to them
  - whereas with class modeling you identify classes and assign responsibilities to them
- Then you identify the **associations** between entities
  - relationships, inheritance, and composition
  - similar to the associations between classes

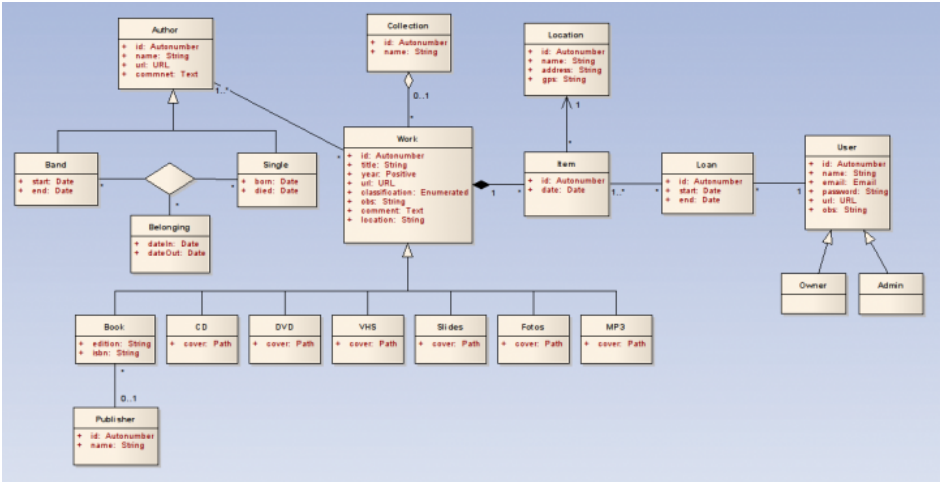
Scott Ambler. The Object Primer. Cambridge University Press, 3rd Edition, 2004. Section. 8.5

CREATE A LDM

- Much of the advice for creating conceptual class diagrams is applicable to creating LDMs
- To create a LDM, **iteratively** perform the following steps:
  1. Identify entity types (a collection of people, places, things, events, or concepts)
  2. Identify relationships (entities have relationships with other entities)
  3. Identify attributes (each entity type will have one or more data attributes)
  4. Apply naming conventions (team standards and guidelines applicable to data model)

Scott Ambler. The Object Primer. Cambridge University Press, 3rd Edition, 2004. Section. 8.5

EXAMPLE: MEDIALIBRARY CONCEPTUAL MODEL



⇒ https://web.fe.up.pt/~jlopes/doku.php/teach/lbaw/medialib/a04

WHAT WE LEARNED

- The Conceptual (Domain) Model describes the domain (of the problem)
- A Conceptual (Data) Model[1] contains the concepts of the domain that must persist
- These conceptual models are similar
- A UML class diagram is used for both

[1] Logical Data Model [Ambler]

Part II

Relational schema (A5)

CONTENTS

⇒ `http://web.fe.up.pt/~jlopes/doku.php/teach/lbaw/artefacts/a05`

RELATIONAL SCHEMA (A5)

A5

This artefact contains the Relational Schema obtained by mapping the Conceptual Data Model.

- The **Relational Schema** includes the relation schemas, attributes, domains, primary keys, foreign keys and other integrity rules: UNIQUE, DEFAULT, NOT NULL, CHECK.

RELATIONAL SCHEMA (A5)

A5

This artefact contains the Relational Schema obtained by mapping the Conceptual Data Model.

- The **Relational Schema** includes the relation schemas, attributes, domains, primary keys, foreign keys and other integrity rules: UNIQUE, DEFAULT, NOT NULL, CHECK.

RELATION SCHEMAS

- Relation schemas are specified in the compact notation:

Table1(id1, attribute1 NN)
Table2(id1, attribute1 → Table1 NN)
Table3(id1, id2 → Table2, attribute1 UK NN)
Table4((id1, id2) → Table3, id3 attribute1)

(UK means UNIQUE and NN means NOT NULL)

- The specification of additional domains:

Today	DATE DEFAULT CURRENT_DATE
Priority	ENUM ('High', 'Medium', 'Low')

In PostgreSQL use lower case and the "snake_case" convention!

RELATION SCHEMAS

- Relation schemas are specified in the compact notation:

Table1(id1, attribute1 NN)
Table2(id1, attribute1 → Table1 NN)
Table3(id1, id2 → Table2, attribute1 UK NN)
Table4((id1, id2) → Table3, id3 attribute1)

(UK means UNIQUE and NN means NOT NULL)

- The specification of additional domains:

Today	DATE DEFAULT CURRENT_DATE
Priority	ENUM ('High', 'Medium', 'Low')

In PostgreSQL use lower case and the "snake_case" convention!

RELATION SCHEMAS

- Relation schemas are specified in the compact notation:

Table1(id1, attribute1 NN)
Table2(id1, attribute1 → Table1 NN)
Table3(id1, id2 → Table2, attribute1 UK NN)
Table4((id1, id2) → Table3, id3 attribute1)

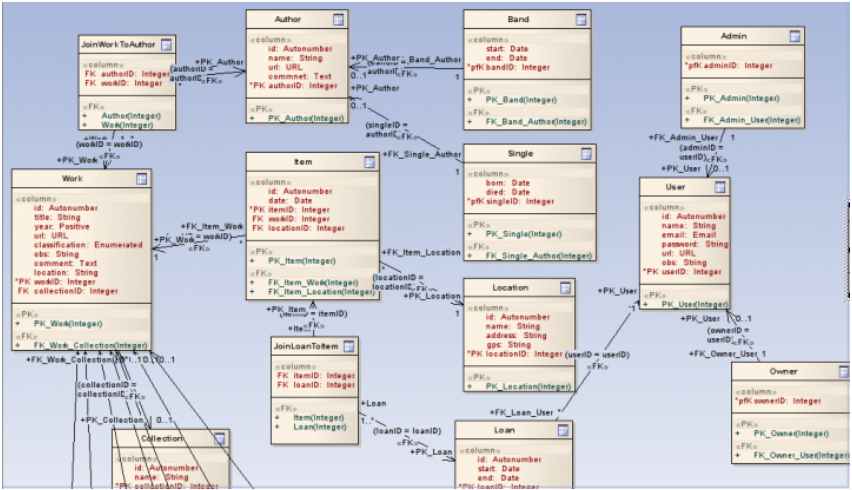
(UK means UNIQUE and NN means NOT NULL)

- The specification of additional domains:

Today	DATE DEFAULT CURRENT_DATE
Priority	ENUM ('High', 'Medium', 'Low')

In PostgreSQL use lower case and the "snake_case" convention!

PHYSICAL DATA MODEL (NOT REQUIRED)



E/A may be used to generate the PDM (but there's a lot of work to clean it afterwards!).

RELATIONAL VS. O-O PARADIGMS

- Tables are the database equivalent of classes; data are stored in physical tables
- A good start is to do a one-to-one mapping of your classes to data tables
- A column is the database equivalent of an attribute, and each table will have one or more columns
- Unlike attributes in classes, which can be either primitive types or other objects, a column may only be a primitive type such (char, int, float)[2]
- There are relationships between tables just like there are relationships between classes
- A stored procedure is conceptually similar to a global method implemented by the database

Scott Ambler. The Object Primer. Cambridge University Press, 3rd Edition, 2004. Section. 12.3

[2]Not true in today's Object-relational DBMS (e.g. PostgreSQL).

MAPPING RULES

UML – Metodologias e Ferramentas CASE, Vol. 1, 2ª Edição, pp. 314-315
Alberto Silva e Carlos Videira, Centro Atlântico (2005)

Regra 1	Classes são mapeadas em esquemas relacionais
Regra 2	Atributos de classes são mapeados em atributos de relações.
Regra 3	Operações de classes não são em geral mapeadas. Podem ser todavia mapeadas em *stored procedures*, guardadas e executadas no contexto global da base de dados envolvida.
Regra 4	Objectos são mapeados em tuplos de uma ou mais relações.
Regra 5	Cada objecto é identificado univocamente. Caso a identificação de um objecto seja definida **explicitamente** pelo estereótipo OID (*object identifier*), associado a um ou mais atributos, esse atributo é mapeado para chave primária no esquema relacional. Caso contrário, assume-se **implicitamente** que a chave primária correspondente é derivada a partir de um novo atributo com o nome da relação e sufixo comum (e.g., "PK", "ID").
Regra 6:	O mapeamento de associações de muitos-para-muitos implica a criação de um novo esquema relacional cujos atributos em conjunto desempenham o papel de chave-primária, e individualmente desempenham o papel de chave-estrangeira para cada um dos esquemas derivados das classes envolvidas.
Regra 7:	O mapeamento de associações de um-para-muitos implica a introdução, no esquema relacional correspondente à classe que tem a restrição "muitos", de um atributo chave-estrangeira para o outro esquema.
Regra 8:	O mapeamento de associações de um-para-um apresenta em geral duas soluções. A primeira, corresponde à fusão dos atributos das classes envolvidas num único esquema comum. A segunda solução corresponde a mapear cada uma das classes em esquemas correspondentes e em escolher um dos esquemas como o mais adequado para a introdução de um atributo chave-estrangeira para o outro esquema. Este atributo deverá ainda ser definido como único no âmbito desse esquema.

MAPPING RULES (2)

Regra 9:	O adorno de navegação não tem em geral qualquer impacto no processo de mapeamento. A excepção reside em associações de um-para-um, que quando são complementadas pelo adorno de navegação ajudam a seleccionar o esquema que deverá incluir o atributo chave-estrangeira.
Regra 10:	O adorno de agregação (simples e composta) tem um impacto mínimo no processo de mapeamento, podendo corresponder à definição de restrições em cascata ("CASCADE") nas operações de alteração e/ou remoção de tuplos.
Regra 11:	O mapeamento de relações de generalização apresenta em geral três soluções. A primeira solução consiste no esmagamento das classes da hierarquia num único esquema correspondente à superclasse original: esta solução é adequada quando não existe uma distinção significativa ao nível da estrutura das sub-classes e/ou quando a semântica da sua identificação não é forte. A segunda solução consiste em considerar apenas esquemas correspondentes às sub-classes e duplicar os atributos da super-classe nesses esquemas; funciona em particular se a super-classe for definida como abstracta. A terceira solução consiste em considerar todos os esquemas, correspondentes a todas as classes da hierarquia, tendo como resultado uma malha de esquemas ligados e mantidos à custa de regras de integridade referencial. Esta solução tem a vantagem de evitar a duplicação de informação entre diferentes esquemas, mas sugere uma pulverização dessa informação por vários esquemas e pode implicar uma penalização de desempenho em operações de consulta ou actualização de dados ao exigir a realização de várias operações de junção (i.e., "JOIN") e/ou de validação de integridade referencial.

Alberto Manuel Rodrigues da Silva e Carlos Alberto Escaleira Videira, UML, metodologias e ferramentas CASE, 2ª Edição, Volume 1, Centro Atlântico Editora, Maio 2005.

WHAT WE LEARNED

- What is a Relational schema
- How to map a Conceptual model in a Relational schema

Part III

SCHEMA VALIDATION (A5)

CONTENTS

⇒ http://web.fe.up.pt/~jlopes/doku.php/teach/lbaw/artefacts/a05

RELATIONAL SCHEMA VALIDATION AND REFINEMENT (A5)

A5

This artefact contains the Relational Schema obtained by mapping the Conceptual Data Model.

- To validate the Relational Schema obtained from the Conceptual Model, all **functional dependencies** are identified and the normalization of all relation schemas is accomplished.
- Should it be necessary, in case the scheme is not in the Boyce–Codd Normal Form (BCNF), the relational schema is refined using **normalisation**.

RELATIONAL SCHEMA VALIDATION AND REFINEMENT (A5)

A5

This artefact contains the Relational Schema obtained by mapping the Conceptual Data Model.

- To validate the Relational Schema obtained from the Conceptual Model, all **functional dependencies** are identified and the normalization of all relation schemas is accomplished.
- Should it be necessary, in case the scheme is not in the Boyce–Codd Normal Form (BCNF), the relational schema is refined using **normalisation**.

THE EVILS OF REDUNDANCY

- **Redundancy** is at the root of several problems associated with relational schemas:
  - redundant storage
  - insert/delete/update anomalies
- Integrity constraints, in particular **functional dependencies**, can be used to identify schemas with such problems and to suggest refinements
- Main refinement technique: **decomposition** (replacing ABCD with, say, AB and BCD, or ACD and ABD)
- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - What problems (if any) does the decomposition cause?

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

THE EVILS OF REDUNDANCY

- **Redundancy** is at the root of several problems associated with relational schemas:
  - redundant storage
  - insert/delete/update anomalies
- Integrity constraints, in particular **functional dependencies**, can be used to identify schemas with such problems and to suggest refinements
- Main refinement technique: **decomposition** (replacing ABCD with, say, AB and BCD, or ACD and ABD)
- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - What problems (if any) does the decomposition cause?

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

FUNCTIONAL DEPENDENCIES (RECAP)

- A functional dependency $X \Rightarrow Y$ holds over relation R if, for every allowable instance r of R:
  - $t1 \in r, t2 \in r, \Pi_X(t1) = \Pi_X(t2) \Rightarrow \Pi_Y(t1) = \Pi_Y(t2)$
  - i.e., given two tuples in r, if the X values agree, then the Y values must also agree (X and Y are sets of attributes)
- An FD is a statement about **all** allowable relations
  - Must be identified based on **semantics of application**
  - Given some allowable instance r1 of R, we can check if it violates some FD f, but we cannot tell if f holds over R!
- K is a candidate key for R means that $K \Rightarrow R$
  - However, $K \Rightarrow R$ does not require K to be minimal!

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

EXAMPLE: CONSTRAINTS ON ENTITY SET

- Consider the relation obtained from entity Hourly_Emps: Hourly_Emps (ssn, name, lot, rating, hrly_wages, hrs_worked)
- **Notation**: We will denote this relation schema by listing the attributes: SNLRWH
  - This is really the **set** of attributes {S,N,L,R,W,H}
  - Sometimes, we will refer to all attributes of a relation by using the relation name. (e.g., Hourly_Emps for SNLRWH)
- Some FDs on Hourly_Emps:
  - **ssn is the key**: $S \Rightarrow SNLRWH$
  - **rating determines hrly_wages**: $R \Rightarrow W$

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

EXAMPLE (CONT.)

Hourly_Emps

ssn	name	lot	rating	hrly_wages	hrs_worked
123-22-3666	Attisboo	48	8	10	40
123-33-3666	Smiley	22	8	10	30
123-44-3666	Smethurst	35	5	7	30
123-55-3666	Guldu	36	6	7	32
123-66-3666	Madayan	35	8	10	40

- Problems due to $R \Rightarrow W$:
  - Update anomaly
    - change wage in just the 1st tuple
  - Insertion anomaly
    - insert an employee and don't know the hourly wage for his rating
  - Deletion anomaly
    - delete all employees with rating 5
  - The use of *null* values does not solve the problemes

EXAMPLE (CONT.)

Hourly_Emps

ssn	name	lot	rating	hrly_wages	hrs_worked
123-22-3666	Attisboo	48	8	10	40
123-33-3666	Smiley	22	8	10	30
123-44-3666	Smethurst	35	5	7	30
123-55-3666	Guldu	36	6	7	32
123-66-3666	Madayan	35	8	10	40

- Problems due to $R \Rightarrow W$:
  - Update anomaly
    - change wage in just the 1st tuple
  - Insertion anomaly
    - insert an employee and don't know the hourly wage for his rating
  - Deletion anomaly
    - delete all employees with rating 5
  - The use of *null* values does not solve the problemes

EXAMPLE (CONT.)

Hourly_Emps

ssn	name	lot	rating	hrly_wages	hrs_worked
123-22-3666	Attisboo	48	8	10	40
123-33-3666	Smiley	22	8	10	30
123-44-3666	Smethurst	35	5	7	30
123-55-3666	Guldu	36	6	7	32
123-66-3666	Madayan	35	8	10	40

- Problems due to $R \Rightarrow W$:
  - Update anomaly
    - change wage in just the 1st tuple
  - Insertion anomaly
    - insert an employee and don't know the hourly wage for his rating
  - Deletion anomaly
    - delete all employees with rating 5
  - The use of *null* values does not solve the problemes

EXAMPLE (CONT.)

Hourly_Emps

ssn	name	lot	rating	hrly_wages	hrs_worked
123-22-3666	Attisboo	48	8	10	40
123-33-3666	Smiley	22	8	10	30
123-44-3666	Smethurst	35	5	7	30
123-55-3666	Guldu	36	6	7	32
123-66-3666	Madayan	35	8	10	40

- Problems due to $R \Rightarrow W$:
  - Update anomaly
    - change wage in just the 1st tuple
  - Insertion anomaly
    - insert an employee and don't know the hourly wage for his rating
  - Deletion anomaly
    - delete all employees with rating 5
  - The use of *null* values does not solve the problemes

EXAMPLE (CONT.)

Hourly_Emps

ssn	name	lot	rating	hrly_wages	hrs_worked
123-22-3666	Attisboo	48	8	10	40
123-33-3666	Smiley	22	8	10	30
123-44-3666	Smethurst	35	5	7	30
123-55-3666	Guldu	36	6	7	32
123-66-3666	Madayan	35	8	10	40

- Problems due to $R \Rightarrow W$:
  - Update anomaly
    - change wage in just the 1st tuple
  - Insertion anomaly
    - insert an employee and don't know the hourly wage for his rating
  - Deletion anomaly
    - delete all employees with rating 5
  - The use of *null* values does not solve the problemes

EXAMPLE (CONT.)

Hourly_Emps2

ssn	name	lot	rating	hrs_worked
123-22-3666	Attisboo	48	8	40
123-33-3666	Smiley	22	8	30
123-44-3666	Smethurst	35	5	30
123-55-3666	Guldu	36	6	32
123-66-3666	Madayan	35	8	40

Wages

rating	wages
5	7
6	7
8	10

- No redundancy problems now!

EXAMPLE (CONT.)

Hourly_Emps2

ssn	name	lot	rating	hrs_worked
123-22-3666	Attisboo	48	8	40
123-33-3666	Smiley	22	8	30
123-44-3666	Smethurst	35	5	30
123-55-3666	Guldu	36	6	32
123-66-3666	Madayan	35	8	40

Wages

rating	wages
5	7
6	7
8	10

- No redundancy problems now!

NORMAL FORMS (RECAP)

- Returning to the issue of schema refinement, the first question to ask is whether any refinement is needed!
- If a relation is in a certain **normal form** (BCNF, 3NF, etc.), it is known that certain kinds of problems are avoided/minimised
  - This can be used to help us decide whether decomposing the relation will help
- Role of FDs in detecting redundancy:
  - Consider a relation R with 3 attributes, ABC
  - **No FDs hold**: There is no redundancy here
  - **Given** $A \Rightarrow B$: Several tuples could have the same A value, and if so, they'll all have the same B value!

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

BOYCE-CODD NORMAL FORM (BCNF)

- Relation R with FDs F is in **BCNF** if, for all $X \Rightarrow A$ in F^+ (wherein A is a single attribute)
  - $A \in X$ ($X \Rightarrow A$ is called a trivial FD), or
  - X contains a key for R
- In other words, R is in BCNF if the only non-trivial FDs that hold over R are key constraints (calculation?)
- **BCNF ensures absence of redundancy in R due to Functional Dependencies**

THIRD NORMAL FORM (3NF)

- Relation R with FDs F is in **3NF** if, for all $X \Rightarrow A$ in F+
  - $A \in X$ (called a trivial FD), or
  - X contains a key for R, or
  - A is part of some key for R
- Minimality of a key is crucial in third condition above!

- If R is in BCNF, obviously in 3NF
- If R is in 3NF, some redundancy is possible
  - It is a compromise, used when BCNF is not achievable (e.g., no "good" decomposition, or performance considerations)

- *Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations always possible*

THIRD NORMAL FORM (3NF)

- Relation R with FDs F is in **3NF** if, for all $X \Rightarrow A$ in F+
  - $A \in X$ (called a trivial FD), or
  - X contains a key for R, or
  - A is part of some key for R
- Minimality of a key is crucial in third condition above!

- If R is in BCNF, obviously in 3NF
- If R is in 3NF, some redundancy is possible
  - It is a compromise, used when BCNF is not achievable (e.g., no "good" decomposition, or performance considerations)

- *Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations always possible*

THIRD NORMAL FORM (3NF)

- Relation R with FDs F is in **3NF** if, for all $X \Rightarrow A$ in F+
  - $A \in X$ (called a trivial FD), or
  - X contains a key for R, or
  - A is part of some key for R
- Minimality of a key is crucial in third condition above!

- If R is in BCNF, obviously in 3NF
- If R is in 3NF, some redundancy is possible
  - It is a compromise, used when BCNF is not achievable (e.g., no "good" decomposition, or performance considerations)

- *Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations always possible*

DECOMPOSITION OF A RELATION SCHEME

- Suppose that relation R contains attributes A1 ... An
- A **decomposition** of R consists of replacing R by two or more relations R1 ... Rm such that:
  - Each new relation scheme contains a subset of the attributes of R (and no attributes that do not appear in R), and
  - Every attribute of R appears as an attribute of one of the new relations
- Intuitively, decomposing R means we will store instances of the relation schemes produced by the decomposition R1 ... Rm, instead of instances of R
  - R1 tuples are obtained by projection of tuples in R of R1 attributes
- E.g. Can decompose **SNLRWH** into **SNLRH** and **RW** (due to $R \Rightarrow W$)

PROBLEMS WITH DECOMPOSITIONS

- There are three potential problems to consider:
  1. Some queries become more expensive
     - e.g., How much did sailor Joe earn? (salary = W*H)
  2. Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
     - Fortunately, not in the SNLRWH example
  3. Checking some dependencies may require joining the instances of the decomposed relations
     - Fortunately, not in the SNLRWH example
- **Tradeoff**: Must consider these issues vs. redundancy

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

PROBLEMS WITH DECOMPOSITIONS

- There are three potential problems to consider:
  1. Some queries become more expensive
     - e.g., How much did sailor Joe earn? (salary = W*H)
  2. Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
     - Fortunately, not in the SNLRWH example
  3. Checking some dependencies may require joining the instances of the decomposed relations
     - Fortunately, not in the SNLRWH example
- **Tradeoff**: Must consider these issues vs. redundancy

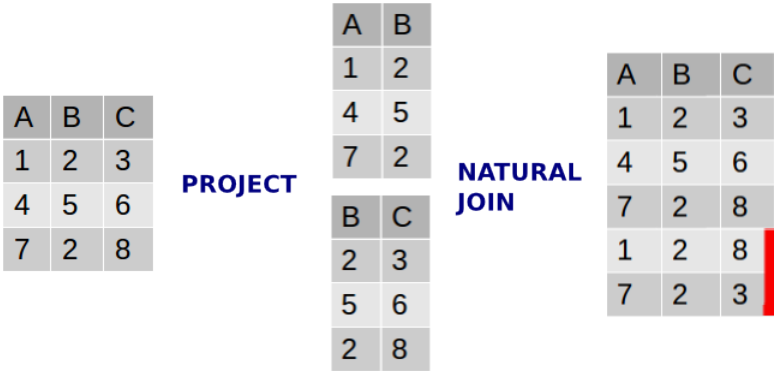
R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

LOSSLESS-JOIN DECOMPOSITIONS

- Decomposition of R into X and Y is **lossless-join** w.r.t. a set of FDs F if, for every instance r that satisfies F:
  - $\Pi_X(r) \bowtie \Pi_Y(r) = r$
- It is always true that $r \subseteq \Pi_X(r) \bowtie \Pi_Y(r)$
  - In general, the other direction does not hold!
  - If it does, the decomposition is lossless-join
- Definition extended to decomposition into 3 or more relations in a straightforward way
  - **It is essential that all decompositions used to deal with redundancy be lossless!** (Avoids Problem (2))

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

DECOMPOSITION WITH LOSSES



(The decomposition of R into UV and R-V is lossless-join if $U \Rightarrow V$ holds over R)

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

EXERCISE

Hourly_Emps2

ssn	name	lot	rating	hrs_worked
123-22-3666	Attisboo	48	8	40
123-33-3666	Smiley	22	8	30
123-44-3666	Smethurst	35	5	30
123-55-3666	Guldu	36	6	32
123-66-3666	Madayan	35	8	40

Wages

rating	wages
5	7
6	7
8	10

- Is this decomposition lossless-join?

CONTRACTS RELATION

- Relation schema:
  - `contracts(contractid, supplierid, projectid, deptid, partid, qty, value)`
  - CSJDPQV
- Integrity constraints that are known to hold:
  - The contract id C a key: $C \Rightarrow CSJDPQV$
  - A project purchases a given part using a single contract: $JP \Rightarrow C$
  - A department purchases at most one part from a supplier: $SD \Rightarrow P$

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

DEPENDENCY PRESERVING DECOMPOSITIONS

- Consider CSJDPQV, C is key, $JP \Rightarrow C$ and $SD \Rightarrow P$
  - BCNF decomposition: CSJDQV and SDP
  - Problem: Checking $JP \Rightarrow C$ requires a join!
- Dependency preserving decomposition (Intuitive):
  - If R is decomposed into X, Y and Z, and we enforce the FDs that hold on X, on Y and on Z, then all FDs that were given to hold on R must also hold. (Avoids Problem (3))
- Projection of set of FDs F:
  - If R is decomposed into X, ... projection of F onto X (denoted F_X) is the set of FDs $U \Rightarrow V$ in F^+ (closure of F) such that U, V are in X

R. Ramakrishnan, J. Gehrke. Schema Refinement and Normal Forms. In Database Management Systems. McGRAW-Hill International Editions, 3rd Edition, 2003, Chapter 19.

DEPENDENCY PRESERVING DECOMPOSITIONS (2)

- A decomposition that preserves dependencies enables us to guarantee all DF examining only one instance of relation in each insertion (or removal) of a tuple
- The decomposition of R, with DF *F*, in relations with sets of attributes X and Y **preserves dependencies** if:
  - $(F_X \cup F_Y)^+ = F^+$
- We need to calculate the closure of the union and compare with the closure of F (**not just F**)
- Dependency preserving does not imply lossless-join
  - ABC, $A \Rightarrow B$, decomposed into AB and BC
- and vice-versa

RELATIONAL SCHEME REFINEMENT

1. Identify the applicable FDs, according to the application semantics (for each relation!)
2. Reduce the FDs set obtained (ie, get the **minimum coverage**)
3. Check the normalisation of relations
4. If the relation is not in BCNF, and there are no other impediments (efficiency reasons, etc.), search for the various possible decompositions (lossless) into relations in BCNF and choose the “best decomposition” (which preserves the greater number of dependencies, it makes more sense, etc.)
5. If there is no satisfactory decomposition to BCNF, and if the relation is no longer in 3NF, consider the lossless decomposition for 3NF, preserving the FDs

SUMMARY OF DATABASE TUNING

- The Relational schema should be refined by considering performance criteria and workload:
  - May choose 3NF or lower normal form over BCNF
  - May choose among alternative decompositions into BCNF (or 3NF) based upon the workload
  - May denormalise, or undo some decompositions
  - May decompose a BCNF relation further!
  - May choose a horizontal decomposition of a relation
- ...

WHAT WE LEARNED

- After mapping the Conceptual model to a Relational Schema, a validation step must take place to make sure there's no redundancy
- Normalisation is the way to refine the schema in order to remove redundancy
- It is essential that all decompositions used to deal with redundancy be lossless
- Sometimes it is better to allow some degree of redundancy in order preserve all dependencies (trade-off)