

Universidade da Beira Interior

Departamento de Informática

Bases de Dados
2018/2019

H. Proença, J. Muranho, P. Prata

Notas de apoio às aulas.

Versão 3.7, 2019-05-15

Ficha da UC (LEI, IW)

Unidade curricular /Curricular Unit:

Bases de Dados / Databases

Docente responsável (preencher o nome completo) e respetivas horas de contacto na unidade curricular:
--

Teacher in charge (fill in the full name) and number of contact hours in the curricular unit:
--

Outros docentes e respetivas horas de contacto na unidade curricular:
--

Other teachers and number of contact hours in the curricular unit:

Objetivos de aprendizagem (conhecimentos, aptidões e competências a desenvolver pelos estudantes):

Esta unidade curricular introduz a temática da conceção, análise e construção de bases de dados relacionais. O seu objetivo principal é, portanto, preparar os alunos para entender, projetar e desenvolver sistemas de bases de dados.

A unidade curricular centra-se no modelo relacional, incidindo em especial sobre modelação, normalização, linguagens de interrogação (álgebra relacional e SQL), gestão da base de dados e aplicações cliente/servidor.

Com a concretização do processo ensino-aprendizagem, o estudante deve ser capaz de:

- Dada uma situação real, ou hipotética, desenvolver um modelo de dados que a represente;
- Normalizar (3FN, BCNF ou superior) e “desnormalizar” as relações;
- Escolher um sistema de gestão de bases de dados em função do sistema de informação a desenvolver;
- Produzir o modelo físico da base de dados;
- Interrogar a base de dados (via SQL);
- Desenvolver aplicações multiutilizador sobre bases de dados cliente/servidor;
- Usar transações.

Intended learning outcomes (knowledge, skills and competences to be developed by the students):
--

This course introduces the theme of design, analysis and construction of the relational paradigm. Therefore, its main objective is to prepare students to understand, design and develop database systems.

The course focuses on the relational model, namely, modelling, normalization, query languages (relational algebra and SQL), database management issues and developing client/server database applications.

Upon completion of the teaching-learning process, the student should be able to:

- Given a real, or hypothetical case, develop a suitable data model;

- Normalize (3NF, BCNF, or a superior normal form) and "de-normalize" relations;
- Choose a database management system that fulfills the needs of the information system to be developed;
- Produce the physical database model;
- Query the database (using SQL);
- Develop multi-user database applications;
- Use transactions.

Conteúdos programáticos:

1. Introdução às bases de dados
 - 1.1 Sistemas de ficheiros vs. Bases de dados “Desktop” vs. Bases de dados cliente/servidor: vantagens, desvantagens e quando usar (ou não usar)
 - 1.2 Conceitos fundamentais
 - 1.3 Modelos de dados (Hierárquico, Rede e Relacional. Estruturas de dados e linguagens de manipulação associadas)
2. Modelo Relacional
 - 2.1 O modelo de dados
 - 2.2 Álgebra relacional
 - 2.3 Linguagens relacionais
 - 2.4 Restrições de integridade
 - 2.5 Dependências lógicas
3. Elaboração do modelo conceptual de uma base de dados
 - 3.1 Modelo entidade-associação
 - 3.2 Teoria da normalização
4. Desenvolvimento de aplicações Cliente/Servidor.
5. Transações
 - 5.1 Propriedades ACID
 - 5.2 Isolamento e fenómenos associados
 - 5.3 Execução concorrente

Syllabus:

1. Introduction to database systems
 - 1.1 Data files vs Desktop databases vs Client/server databases: advantages, disadvantages and when use (or not use)
 - 1.2 Fundamental concepts
 - 1.3 Data models (hierarchic; network; and relational. Data structures and manipulation languages)
2. The relational model.
 - 2.1 The data model
 - 2.2 Relational algebra
 - 2.3 Database query languages
 - 2.4 Integrity constraints
 - 2.5 Logical dependences

- 3. Conceptual database analysis and design
 - 3.1 Entity-Relationship modelling
 - 3.2 Normalization
- 4. Client/Server applications development
- 5. Transactions
 - 5.1 ACID properties
 - 5.2 Transaction isolation and related phenomena
 - 5.3 Concurrency

Demonstração da coerência dos conteúdos programáticos com os objetivos de aprendizagem da unidade curricular:

O desenvolvimento dos conteúdos programáticos é centrado no objetivo de preparar os estudantes para entender, projetar e desenvolver soluções segundo o modelo relacional. Numa primeira fase são apresentados os conceitos gerais e as soluções baseadas em bases de dados (BD) desktop e em BD cliente/servidor e é realçado o papel da nova entidade: o sistema de gestão de bases de dados. De seguida, são apresentados os modelos de dados hierárquico, rede e relacional, sendo o enfoque colocado neste último. Assimilados os conceitos e conhecidas as linguagens relacionais, passa-se para a produção de modelos de dados dotados de boas propriedades. São, então, trabalhadas as técnicas para a produção do modelo entidade-associação (DEA) e do respetivo esquema relacional, e é estudada a temática da normalização e analisado o refinamento de modelos de dados existentes.

Evidence of the syllabus coherence with the curricular unit's intended learning outcomes:

The syllabus is focused on the course main objective: prepare the students to understand, design and develop relational databases. Thus, initially are introduced the general concepts and the solutions based on the desktop and client/server databases. The role of the Database Management System is then highlighted. Then the focus is put on the data models (hierarchical, network and relational). The relational model is then studied in detail.

Assimilated the relational concepts and known the relational languages, the attention is given to produce data models with good properties. At this point, the techniques to build entity-relationship diagrams and its relational schema are studied. The normalization of relations is studied and applied to the refine existing data models.

Metodologias de ensino (avaliação incluída):

As aulas estão organizadas em aulas teóricas (T), para exposição dos conteúdos programáticos (diapositivos e escrita manual) e para interação com os alunos, e aulas práticas (PL), em salas devidamente equipadas, onde se exemplificam e exploram cenários concretos de utilização dos diversos tipos de bases de dados (desktop – MS ACCESS e cliente/servidor – MS SQL Server), se resolvem exercícios práticos sobre os assuntos abordados no programa e onde se dá continuidade à execução do trabalho prático.

Teaching methodologies (including assessment):

The course is structured with alternated theoretical (T) classes, for syllabus exposure and interaction with students, and practical classes (PL), to explore and exemplify concrete scenarios of application of different types of databases (desktop - MS ACCESS and client/server - MS SQL server) and solve exercises about all topics covered in the syllabus. The practical classes are also used by students to implement the practical work.

Demonstração da coerência das metodologias de ensino com os objetivos de aprendizagem da unidade curricular

A unidade curricular tem a duração de um semestre letivo, envolvendo 60 horas de contacto, 104 horas de trabalho autónomo e 4 horas para avaliação (168 horas no total). As 15 aulas teóricas, de carácter mais expositivo, são usadas para contextualizar as temáticas, introduzir conceitos e desenvolver os temas. Os alunos têm antecipadamente acesso aos diapositivos usados nas aulas, donde podem complementar esse material com as explicações orais apresentadas durante as mesmas.

As aulas práticas decorrem em laboratório com acesso a bases de dados desktop e a servidores de bases de dados cliente/servidor, estando os computadores apetrechados com o software necessário para o desenvolvimento de aplicações. Portanto, durante as aulas práticas, os alunos resolvem exercícios sobre as diferentes temáticas, desenvolvem e exploram diferentes tipos de bases de dados (MS ACCESS e MS SQL Server), interrogam/consultam as bases de dados (via SQL) e desenvolvem aplicações informáticas sobre bases de dados.

Com a execução dos exercícios práticos, os estudantes têm a possibilidade de concretizar, faseadamente, todos os passos inerentes à conceção, análise e construção de uma base de dados e de desenvolver de uma aplicação que interatue sobre a mesma.

Em sumula, a metodologia seguida é adequada e permite atingir os objetivos definidos para a unidade curricular.

Evidence of the teaching methodologies coherence with the curricular unit's intended learning outcomes:

This is a semiannual course, involving 60 hours of contact, 104 hours of autonomous work and 4 hours for evaluation (total: 168 hours).

The 15 theoretical classes, with more expository character, are used to introduce the concepts and develop the themes. The students have access to the accompanying slides in advance, so during the classes they can take notes about the oral explanation of the subjects.

The practical classes take place in a well-equipped laboratory with access to desktop database software and to database servers. The lab computers are prepared with the necessary software for developing database applications. Therefore, in the practical classes, the students solve exercises about the different subjects, develop and explore different kind of databases (MS ACCESS and MS SLQ Server), formulate database queries and develop database applications.

With the practical exercises, the students have the opportunity to implement, in phases, the design and analysis of the database and develop an application that interacts with the developed database.

In short, the methodology is appropriate and achieves the defined objectives for the

course.

Bibliografia principal

1. Thomas Connolly, Carolyn Begg. "Database Systems, A Practical Approach to Design, Implementation and Management", 6th Edition, 2015. Pearson, ISBN: 978-1-292-06118-4.
2. Feliz Gouveia. "Fundamentos de Bases de Dados", FCA, 2014, ISBN: 978-972-722-799-0.
3. Luís Damas, "SQL", 14ª Edição, FCA, 2017, ISBN 978-972-722-829-4.
- 4) R. Ramakrishnan and J. Gehrke, Database Management Systems, McGraw-Hill, 2003.

Main bibliography

1. Thomas Connolly, Carolyn Begg. "Database Systems, A Practical Approach to Design, Implementation and Management", 6th Edition, 2015. Pearson, ISBN: 978-1-292-06118-4.
2. Feliz Gouveia. "Fundamentos de Bases de Dados", FCA, 2014, ISBN: 978-972-722-799-0.
3. Luís Damas, "SQL", 14ª Edição, FCA, 2017, ISBN 978-972-722-829-4.
- 4) R. Ramakrishnan and J. Gehrke, Database Management Systems, McGraw-Hill, 2003.

Ficha UC (MEI)

Unidade curricular /Curricular Unit:

Sistemas de Gestão de Bases de Dados / Database Management Systems
--

Docente responsável (preencher o nome completo) e respetivas horas de contacto na unidade curricular:
--

Teacher in charge (fill in the full name) and number of contact hours in the curricular unit:
--

Outros docentes e respetivas horas de contacto na unidade curricular:
--

Other teachers and number of contact hours in the curricular unit:

Objetivos de aprendizagem (conhecimentos, aptidões e competências a desenvolver pelos estudantes):

Esta Unidade Curricular tem dois objetivos principais: 1) aprofundar os conhecimentos adquiridos na unidade curricular introdutória às “Bases de Dados”, do 1º ciclo de estudos, nomeadamente, aspetos avançados da programação SQL e Tecnologias dos Sistemas de Gestão de Bases de Dados Relacionais; e 2) introduzir a temática das bases de dados não-estruturadas e preparar os alunos para entender, projetar e desenvolver soluções informáticas usando bases de dados NoSQL.

Concluídos os estudos, os estudantes devem conhecer e entender:

- As diferenças entre base de dados relacional e bases de dados não-estruturadas;
- Os conceitos de replicação, distribuição, partição e resiliência;
- Escolher o tipo de base de dados apropriado para uma dada aplicação e prever o seu desempenho quando sujeito a diferentes cargas de dados.

Em resumo, no final, os estudantes terão um entendimento crítico das estratégias e dos problemas associados às bases de dados e serão capazes de propor novas soluções.

Intended learning outcomes (knowledge, skills and competences to be developed by the students):
--

This course has two main goals: 1) consolidate the knowledge acquired in an introductory course of “Databases”, from a first cycle course, particularly, advanced aspects of the SQL programming and the Relational Database Management Systems technologies; and 2) introducing the non-relational databases and preparing students to understand, design and develop computer solutions using NoSQL databases.

Upon completion of the teaching-learning process, the students should know and understand:

- The differences between a relational database and a non-relational database;
- The concepts of replication, distribution, sharding, and resilience;
- How to choose a suitable database for an application and infer its performance when subject to different data overloads.

In resume, after the course, students will have a critical understanding of the strategies and problems associated with the database systems and be able to propose new solutions.

Conteúdos programáticos:

Parte I – Aspetos Avançados de Bases de Dados Estruturadas

1. Modelo relacional
 - 1.1 Sistema de gestão de bases de dados e arquitetura ANSI/SPARC
 - 1.2 Armazenamento de dados
 - 1.3 Indexação
 - 1.4 Processamento e otimização de consultas
 - 1.5 Gestão de transações
 - 1.6 Data warehousing
 - 1.7 Bases de dados temporais

Parte II – Bases de dados não estruturadas (NoSQL)

2. Bases de dados não-estruturadas (NoSQL)
 - 2.1 Contexto e definições
 - 2.2 Motivação
 - 2.3 Taxonomia
3. Distribuição de dados e consistência
 - 3.1 Princípios fundamentais
 - 3.1.1 Modelos de dados flexíveis
 - 3.1.2 Escalabilidade horizontal
 - 3.1.3 Relaxamento da consistência
 - 3.2 Distribuição de dados
 - 3.2.1 Partição
 - 3.2.2 Replicação
 - 3.2.3 Agregação
 - 3.3 Consistência
 - 3.3.1 Consistência na leitura e na escrita
 - 3.3.2 ACID, BASE e CRUD
 - 3.3.3 O Teorema CAP
 - 3.3.4 Relaxamento da consistência
4. Modelos de computação – uma breve introdução
 - 4.1 MapReduce
 - 4.2 Google File System
 - 4.3 Apache Hadoop
- 5 Modelos de Bases de Dados
 - 5.1 Chave-Valor
 - 5.2 Orientado a Documentos
 - 5.3 Orientado a Colunas

Syllabus:

Part I – Advanced aspects of structured databases

1. Relational Model

1.1 ANSI/SPARC architecture and database management systems

1.2 Data storage

1.3 Indexing

1.4 Query processing and optimization

1.5 Transaction management

1.6 Data warehousing

1.7 Temporal databases

Part II – Unstructured Databases (NoSQL)

2. Unstructured Databases (NoSQL)

2.1 Context and Definitions

2.2 Motivation

2.3 Taxonomy

3. Data distribution and consistency

3.1 Fundamental principles

3.1.1 Flexible data models

3.1.2 Horizontal scalability

3.1.3 Relaxation of consistency

3.2 Distribution of data

3.2.1 Partitioning

3.2.2 Replication

3.2.3 Aggregation

3.3 Consistence

3.3.1 Consistence on reading and writing

3.3.2 ACID, BASE and CRUD

3.3.3 The CAP Theorem

3.3.4 Relaxation of consistency

4. Computing Models– a brief introduction

4.1 MapReduce

4.2 Google File System

4.3 Apache Hadoop

5 Database Models

5.1 Key-Value

5.2 Document oriented

5.3 Column oriented

5.4 Graph oriented

Demonstração da coerência dos conteúdos programáticos com os objetivos de aprendizagem da unidade curricular:

O capítulo 1 da Parte I dos conteúdos programáticos incide sobre o modelo relacional e centra-se no estudo das tecnologias tipicamente implementadas num Sistema de Gestão de Bases de Dados Relacional para armazenar e pesquisar de forma eficiente grandes quantidades de dados. Nesse sentido são também abordados os temas da indexação e processamento e otimização de consultas. Apresentam-se ainda a temática do controlo de transações e recuperação de falhas e o problema da escalabilidade, atingindo-se assim o primeiro grande objetivo da Unidade Curricular.

O segundo objetivo principal, ou seja, introduzir a temática das bases de dados não estruturadas e preparar os alunos para entender, projetar e desenvolver soluções informáticas usando bases de dados NoSQL, é trabalhado no restante programa curricular, onde se apresentam os diferentes modelos de bases de dados não-estruturadas e a sua problemática.

Evidence of the syllabus coherence with the curricular unit's intended learning outcomes:

Chapter 1 of Part I of the syllabus focuses on the relational model and is related with the study of technologies typically implemented on a Relational Database Management System to efficiently store and search large amounts of data. Over this, the topics of indexing and processing and optimization of queries are also addressed. This chapter also includes the subjects of transactions control and scalability problems. So, the first main goal is fulfilled.

The second main goal, that is, introduction the unstructured databases and preparing students to understand, design, and develop computer solutions using NoSQL databases, is worked on the rest of the syllabus, where the different models of non-relational databases are studied as so its related problems.

Metodologias de ensino (avaliação incluída):

As aulas estão organizadas em aulas teóricas (T), para exposição dos conteúdos programáticos (diapositivos e escrita manual) e para interação com os alunos, e aulas práticas (PL), em salas devidamente equipadas, onde se exemplificam e exploram cenários concretos de utilização dos diversos tipos de bases de dados relacionais (MS SQL Server) e não-relacionais (Riak, MongoDB, CouchDB, Cassandra, Neo4j, entre outros), se resolvem exercícios práticos sobre os assuntos abordados no programa e onde se dá continuidade à execução dos trabalhos práticos. Nas aulas decorrem também apresentações dos temas tratados pelos alunos.

Os trabalhos práticos (projetos) são desenvolvidos em grupo.

A avaliação compreende três componentes:

- Parte escrita (10 valores) – um teste a realizar nas últimas semanas de aulas;
- Dois trabalhos práticos (4 valores cada), um sobre transações no SQL Server; e o outro sobre uma base de dados NoSQL;
- Um tema (2 valores), com apresentação, sobre uma base de dados NoSQL.

Teaching methodologies (including assessment):

The course is structured with alternated theoretical (T) classes, for syllabus exposure and interaction with students, and practical classes (PL), to explore and exemplify concrete scenarios of application of different kind of databases relational (MS SQL server) or non-relational (Riak, MongoDB, CouchDB, Cassandra, Neo4j, among others) and solve exercises about all topics covered in the syllabus. The practical classes are also used by students to implement the practical work. The students are required to participate actively in classes, so on theoretical classes also occurs presentations prepared by students.

Practical works (projects) are developed in the group.

The evaluation consists of three components:

- Written test (10 points): one test near the end of the semester;
- Two practical works (4 points, each), one about transactions on SQL Server; and the other about one NoSQL database.
- One theme with oral presentation (2 points), about a NoSQL database model.

Demonstração da coerência das metodologias de ensino com os objetivos de aprendizagem da unidade curricular

A unidade curricular tem a duração de um semestre letivo. As aulas teóricas, de carácter mais expositivo, são usadas para contextualizar as temáticas, introduzir conceitos e desenvolver os temas. Os alunos têm antecipadamente acesso aos diapositivos usados nas aulas, donde podem complementar esse material com as explicações orais apresentadas durante as mesmas. São também fornecidos artigos científicos sobre as matérias apresentadas e que servem também de base aos temas a tratar pelos diferentes grupos de trabalho.

As aulas práticas decorrem em laboratório com acesso a bases de dados cliente/servidor (SQL Server) e NoSQL, estando os computadores apetrechados com o software necessário para o desenvolvimento de aplicações. Portanto, durante as aulas práticas, os alunos resolvem exercícios sobre as diferentes temáticas, desenvolvem e exploram diferentes tipos de bases de dados, interrogam/consultam as bases de dados e desenvolvem aplicações informáticas sobre bases de dados.

Com a execução dos trabalhos práticos, os alunos, para além do trabalho em equipa, têm a possibilidade de concretizar, faseadamente, todos os passos inerentes à conceção, análise e construção de uma base de dados e de desenvolver de uma aplicação que interatue sobre a mesma.

Em sumula, a metodologia seguida é adequada e permite atingir os objetivos definidos para a unidade curricular.

Evidence of the teaching methodologies coherence with the curricular unit's intended learning outcomes:

This is a semiannual course. The theoretical classes, with more expository character, are used to introduce the concepts and develop the subjects. The students have access to the accompanying slides in advance, so during the classes they can take notes about the oral explanation of the subjects. Scientific papers are also provided on certain subjects, so,

students can study in advanced and prepare their themes for oral presentation.

The practical classes take place in a well-equipped laboratory with access to client/server and NoSQL databases. The lab computers are prepared with the necessary software for developing database applications. Therefore, in the practical classes, the students solve exercises about the different subjects, develop and explore different kind of databases, formulate database queries and develop database applications.

With the practical work, the students work as a team and have opportunity to implement, in phases, the design and analysis of the database and develop an application that interacts with the developed database.

In short, the methodology is appropriate and achieves the defined objectives for the course.

Bibliografia principal

- 1) Thomas Connolly, Carolyn Begg. "Database Systems, A Practical Approach to Design, Implementation and Management", 6th Edition, 2015. Pearson, ISBN: 978-1-292-06118-4.
- 2) Sadalage, P. J., & Fowler, M. (2013). "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence". Addison-Wesley Professional, ISBN: 978-0321826626.
- 3) Tiwari, S. (2011). "Professional NoSQL". John Wiley & Sons, Inc., Indianapolis, ISBN: 978-0-470-94334-6.
- 4) Redmond, E. & Wilson, J.R. (2012). "Seven Databases in Seven Weeks. A Guide to Modern Databases and the NoSQL Movement". Pragmatic Bookshelf, ISBN: 978-1-93435-692-0.

Main bibliography

- 1) Thomas Connolly, Carolyn Begg. "Database Systems, A Practical Approach to Design, Implementation and Management", 6th Edition, 2015. Pearson, ISBN: 978-1-292-06118-4.
- 2) Sadalage, P. J., & Fowler, M. (2013). "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence". Addison-Wesley Professional, ISBN: 978-0321826626.
- 3) Tiwari, S. (2011). "Professional NoSQL". John Wiley & Sons, Inc., Indianapolis, ISBN: 978-0-470-94334-6.
- 4) Redmond, E. & Wilson, J.R. (2012). "Seven Databases in Seven Weeks. A Guide to Modern Databases and the NoSQL Movement". Pragmatic Bookshelf, ISBN: 978-1-93435-692-0.

Índice

1	Introdução aos Sistemas de Bases de Dados	1
1.1	Sistemas de Armazenamento de Dados	1
1.1.1	Sistemas de Ficheiros	2
1.1.2	Sistemas de Bases de Dados	5
1.1.2.1	Níveis de abstração.....	6
1.1.2.2	Independência de dados	10
1.1.2.3	Arquitetura ANSI/SPARC	11
1.2	Objetivos e Capacidades de um SGBD	12
1.2.1	Linguagens da base de dados.....	13
1.2.2	Transações	14
1.2.3	Propriedades ACID	15
1.2.4	Controlo de acesso.....	17
1.2.5	Tolerância a Falhas	18
1.2.6	Arquitetura cliente-servidor.....	19
1.3	Processo de desenvolvimento de sistemas de bases de dados	20
1.3.1	Etapas do desenvolvimento de sistemas de bases de dados	20
1.3.2	Atividades principais do desenvolvimento.....	22
1.3.3	Critérios de produção de um modelo de dados ótimo	23
2	Modelos de Dados	24
2.1	Introdução	24
2.1.1	Modelo Hierárquico.....	28
2.1.2	Modelo em Rede.....	31
2.1.3	Modelo Relacional.....	35
2.1.3.1	Álgebra Relacional	39
2.1.3.2	Chaves	44
2.2	Modelo Relacional	47
2.2.1	Estrutura de Dados Relacional	47
2.2.1.1	Modelo Conceptual de Dados	47
2.2.1.2	Entidades, Atributos e Domínios	48
2.2.1.3	Representação de entidades por tuplos.....	49
2.2.1.4	Relação	51
2.2.1.5	Base de dados relacional e esquema relacional.....	51
2.2.2	Álgebra Relacional	53
2.2.2.1	Projeção	54
2.2.2.2	Restrição (ou seleção)	56
2.2.2.3	Operações com conjuntos.....	57
2.2.2.4	Operações de junção.....	59
2.2.2.5	Divisão	65
2.2.2.6	Renomear (rebatizar).....	68
2.2.2.7	Operações de agregação e de agrupamento.....	68
2.2.2.8	Combinação de operações para formar <i>Queries</i>	72

2.3	SGBD relacional	74
2.3.1	Estrutura de um SGBD relacional	74
2.3.2	Processamento de consultas.....	76
2.3.3	Catálogo.....	77
2.3.4	Fases do processamento de consultas	79
2.3.5	Otimização de consultas	81
2.3.6	As doze regras de Codd	85
2.4	Linguagens Relacionais	87
2.4.1	História do SQL:.....	88
2.4.2	Query block	88
2.4.3	Projeção	89
2.4.4	Restrição	90
2.4.5	Junção (Equijunção)	92
2.4.6	Produto Cartesiano	93
2.4.7	União, Intersecção e Diferença.....	93
2.4.8	Divisão.....	94
2.4.9	Funções Standard.....	95
2.4.10	Actualizações.....	96
2.4.10.1	Inserção	96
2.4.10.2	Eliminação.....	97
2.4.10.3	Actualização	97
2.5	Restrições de integridade	98
2.5.1	Integridade de domínio	99
2.5.2	Integridade de entidade.....	99
2.5.3	Integridade referencial	100
2.5.4	Regras de negócio.....	101
3	Teoria da Normalização	102
3.1	Dados redundantes	102
3.2	Dependências Funcionais	104
3.2.1	Definição de Dependência Funcional (DF)	105
3.2.2	Diagrama de Dependência Funcional.....	105
3.2.3	Chave (candidata)	106
3.2.4	Superchave.....	106
3.2.5	Toda a relação tem uma chave	106
3.2.6	Chave Primária	107
3.2.7	Propriedades básicas das DFs.....	107
3.2.7.1	Unicidade	107
3.2.7.2	Reflexibilidade	107
3.2.7.3	Transitividade.....	107
3.2.7.4	Aumento	107
3.2.7.5	Axiomas de Armstrong	107
3.2.8	Propriedades derivadas das DFs	108
3.2.8.1	Distributividade (decomposição)	109
3.2.8.2	União	109
3.2.8.3	Pseudotransitividade.....	109
3.2.9	DF trivial	110
3.2.10	Fecho de um conjunto de DFs	110

3.2.11	Fecho de um conjunto de atributos.....	111
3.2.12	Cobertura e equivalência	112
3.2.13	Chaves e Fecho	114
3.2.14	Perda de informação	114
3.2.15	Decomposição sem perda (<i>Lossless Join</i>)	115
3.3	Normalização	117
3.3.1	Primeira Forma Normal (1FN)	118
3.3.2	Segunda Forma Normal (2FN)	120
3.3.2.1	DF Elementar	121
3.3.2.2	DF Parcial.....	122
3.3.2.3	Segunda Forma Normal (2FN).....	122
3.3.2.4	Casos especiais de relações na 2FN:	123
3.3.2.5	Decomposição em 2FN	124
3.3.3	Terceira Forma Normal (3FN)	125
3.3.3.1	DF Direta.....	127
3.3.3.2	Terceira Forma Normal (3FN)	127
3.3.3.3	3FN e 2FN	128
3.3.3.4	Decomposição em 3FN	128
3.3.4	Forma Normal de Boyce-Codd (FNBC)	130
3.3.4.1	Definição (FNBC)	131
3.3.4.2	FNBC e 3FN.....	135
3.3.4.3	Decomposição em FNBC.....	135
3.3.5	Preservação de dependências.....	136
4	Modelo Entidade–Associação	139
4.1	Introdução	139
4.2	Modelo Entidade-Associação	141
4.3	Propriedades das associações.....	144
4.3.1	Grau de uma associação.	144
4.3.1.1	Associação 1:1.....	144
4.3.1.2	Associação 1:N.....	145
4.3.1.3	Associação M:N.....	146
4.3.2	Tipo de participação	148
4.4	Decomposição de Associações M:N	151
4.5	Associações Complexas.....	155
4.5.1	Grau 1:1:1	156
4.5.2	Grau 1:1:N	157
4.6	Situações Ambíguas.....	158
4.7	Esquema Relacional.....	168
4.7.1	Associações 1:1	168
4.7.2	Associações 1:N	170
4.7.3	Associações M:N.....	171
4.8	Associações Unárias	173
4.8.1	Associações 1:1	173
4.8.2	Associações 1:N	173
4.8.3	Associações M:N.....	174

4.9	Afetação de atributos a “esboços” de esquemas de relação.....	176
4.10	Extensão do esquema relacional	179
4.11	Tabelas supérfluas.....	181
4.12	Subentidades	183
4.13	Conceção final do esquema relacional.....	185
4.14	Exemplo – Biblioteca.....	187
5	Normalização avançada.....	197
5.1	Dependências Multivalor (DM).....	197
5.1.1	Definição 1 (DM)	199
5.1.2	Definição 2 (DM)	200
5.1.3	DFs e DM	200
5.1.4	DM Complementares.....	200
5.1.5	Separação de DM.....	201
5.1.6	Restabelecer uma relação pela junção das suas projeções.	202
5.1.7	DM triviais.....	203
5.2	Dependências de Junção (DJ)	203
5.2.1	Definição	203
5.2.2	Dependências de Junção e Dependências Multivalor.....	204
5.2.3	Dependências de Junção e Dependências baseadas em Chaves.....	205
5.3	Quarta Forma Normal	205
5.3.1	4ª Forma Normal (4FN).....	206
5.3.2	4FN e FNBC.....	206
5.4	Quinta Forma Normal	208
5.4.1	Definição:	208
5.4.2	Normalização em 5FN.....	209
	Referências	210

1 Introdução aos Sistemas de Bases de Dados

Definição:

Uma Base de Dados é uma coleção de dados partilhados, interrelacionados e usados para múltiplos objetivos.

O conceito de base de dados faz hoje parte do nosso dia-a-dia, mesmo que por vezes de forma não explícita.

Exemplos:

Acedemos a bases de dados,

- Quando fazemos compras num hipermercado.
- Quando usamos um cartão de crédito (ou de débito)
- Quando procuramos um livro na biblioteca.
- Quando procuramos um programa de férias numa agência de viagens
- Quando acedemos à página dos serviços académicos

1.1 Sistemas de Armazenamento de Dados

O primeiro sistema de armazenamento automático de dados foi o sistema de ficheiros que usou o mesmo modelo que os sistemas de ficheiros manuais existentes.

Exemplo: fichas dos pacientes num consultório médico.

1.1.1 Sistemas de Ficheiros

Num sistema de ficheiros, cada aplicação cria e mantém os ficheiros com os dados necessários para a sua execução.

Quando surge uma nova aplicação, na maioria dos casos, é necessário criar ficheiros, com campos que provavelmente já existem noutros ficheiros

Problemas

Suponhamos uma organização com centenas de ficheiros...

- **Alto nível de redundância**

Quando duas aplicações que necessitam de determinado item de dados e não “tenham” conhecimento de que este já está registado noutro local ou se este estiver armazenado noutro ficheiro com estrutura desconhecida.

Os mesmos dados podem ser guardados simultaneamente em múltiplos locais.

- **Inconsistência da informação**

As diferentes versões de um item podem estar em diferentes estágios de atualização (conter diferentes valores).

É difícil manter a consistência a assegurar a integridade dos itens de dados.

- **Inflexibilidade**

Um pedido de informação que necessite de dados provenientes de diferentes locais pode não poder ser atendido em tempo útil.

A aplicação pode não controlar todos os recursos necessários.

Mesmo que os dados existam, pode não ser possível construir a informação.

- **Acessos concorrentes**

Diversas aplicações podem partilhar o acesso (leitura / escrita) aos ficheiros necessários para a sua execução.

A inibição de acessos concorrentes pode prejudicar o desempenho das aplicações. Por outro lado, a sua permissão, pode originar inconsistência na informação disponibilizada.

Caso as aplicações não contenham mecanismos de sincronização entre elas, pode ser disponibilizada informação errada.

A implementação de mecanismos de interação pode aumentar consideravelmente a complexidade das aplicações e o tempo necessário para a sua implementação e depuramento.

- Semáforos

- Sockets

...

- **Isolamento e integridade dos dados**

Os dados encontram-se em diferentes ficheiros, cada um com a estrutura e a organização que interessa à aplicação que o criou.

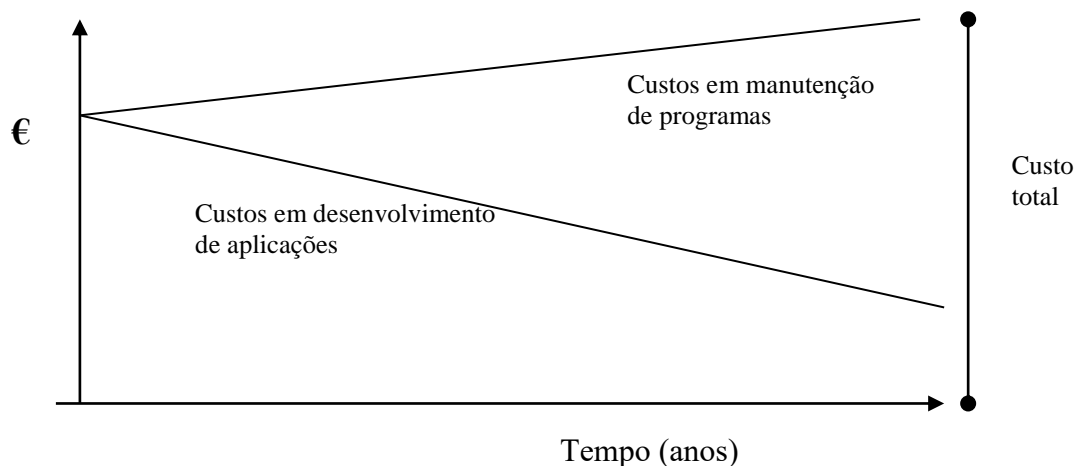
Uma vez que o relacionamento entre os dados é feito ao nível das aplicações, estes permanecem “isolados” em cada componente (ficheiro).

A eliminação ou alteração de parte destes dados por alguma outra aplicação pode facilmente conduzir à perda de integridade da informação.

- **Elevados custos de manutenção**

Cada aplicação que acede a um determinado ficheiro tem de conter uma especificação do respetivo modelo físico e do seu protocolo de acesso. Uma simples alteração nesse ficheiro pode propagar a necessidade de alteração de todas as aplicações que acedem ou registam informação nesse ficheiro.

- *Elevado custo resultante da afetação de pessoal para esse fim.*
- *“Desperdício” de tempo na realização de tarefas que não constituirão qualquer mais-valia para o desempenho da aplicação.*



Um dos principais objetivos de um sistema de base de dados é que um programa possa ser modificado, alterando a forma de utilização dos dados,

sem que isso implique alterações nos restantes programas que utilizam os mesmos dados.

1.1.2 Sistemas de Bases de Dados

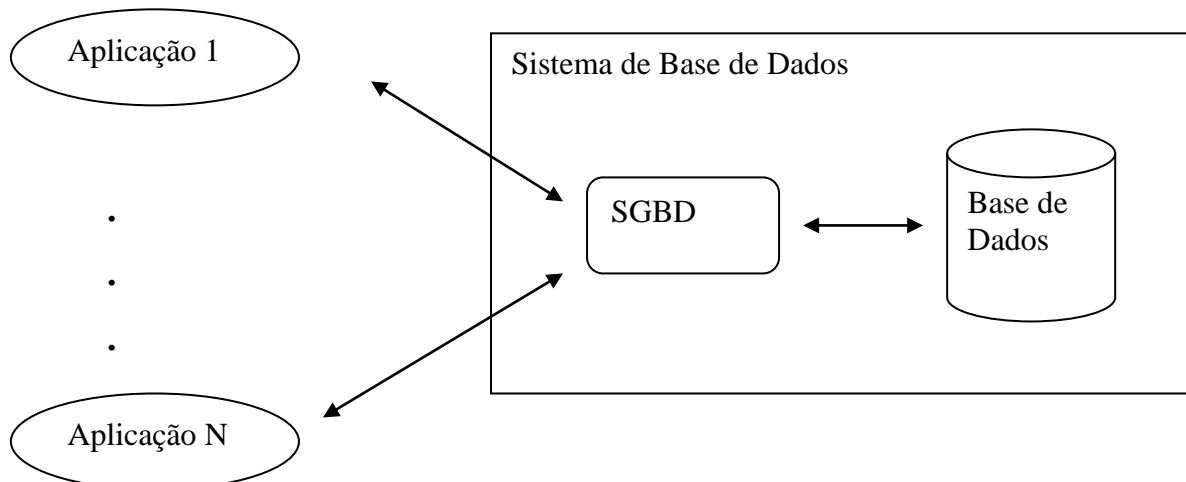
Um sistema de base de dados tenta baixar os custos de manutenção através da separação entre a forma como os dados são percebidos pelo programador e a forma como esses dados são armazenados fisicamente.

Se um programador altera uma estrutura de dados, essa nova estrutura é criada a partir da Base de Dados através do *software* de gestão da Base de Dados e não tem que refletir-se nos outros programas.

=> Existem Registos Lógicos

Cada programa refere-se a registos lógicos de dados e não a registos físicos.

Os dados passam a estar integrados num único conjunto, sendo este administrado por uma aplicação específica: o Sistema de Gestão de Bases de Dados - SGBD.



SGBD – conjunto de programas que permite desempenhar as tarefas de armazenamento e manipulação de dados, fornecendo aos programadores e utilizadores finais os dados tal como eles são pedidos.

O acesso aos dados implica obrigatoriamente a comunicação com uma entidade (SGBD) que reserva para si os privilégios de acesso físico à base de dados e aos ficheiros que a compõem.

Independência dos dados

Independência entre as aplicações e o formato em que é registada a informação. Uma vez que cada aplicação apenas tem de comunicar com o SGBD no processo de consulta e alteração de dados, pode abstrair-se da forma como estes são internamente mantidos.

Ao contrário dos sistemas de ficheiros, é possível que:

“Uma aplicação possa ser modificada, alterando a forma de utilização ou acesso à informação, sem que isso implique alterações nos restantes programas que partilham a utilização da mesma informação”

⇒ Cada aplicação tem a sua estrutura lógica de dados

1.1.2.1 Níveis de abstração

Consideram-se três níveis de abstração:

- Nível interno

Descrição do armazenamento físico da informação numa base de dados.

Definição das estruturas físicas que permitam obter um nível de desempenho, segurança e consistência satisfatório.

Definição das políticas de armazenamento de informação, de acordo com o número, exigência e necessidades de cada cliente específico.

⇒ Coleção de ficheiros, índices e outras estruturas de armazenamento

Internal level ⇒ The physical representation of the database on the computer. This level describes **how** the data is stored in the database.

O nível interno lida com a implementação física da base de dados para alcançar um nível de desempenho e uma utilização de espaço de armazenamento ótimos. Faz a interface com os métodos de acesso do sistema operativo (técnicas de gestão de ficheiros para ler/escrever dados) para armazenar os dados nos dispositivos de armazenamento, criar índices, obter os dados, entre outros. O nível interno tem a seu cargo:

- atribuição (alocação) de espaço de armazenamento para dados e índices;
- descrição dos registos para armazenamento (incluindo tamanho dos itens de dados);
- armazenamento de registos;
- técnicas de compressão e cifragem de dados.

Por baixo do nível interno, existe o nível físico que pode ser gerido pelo sistema operativo sob a direção do SGBD. Contudo, as funções do SBDG e do sistema operativo no nível físico não estão claramente estabelecidas e variam de sistema para sistema. Alguns SGBDs tiram partido dos vários métodos de acesso do sistema operativo, enquanto outros usam somente os mais básicos e criam as suas próprias organizações de ficheiros.

- Nível conceptual

Abstração do mundo real, no que respeita aos utilizadores da base de dados.

O SGBD tem uma linguagem de definição de dados que permite ao utilizador descrever a implementação do esquema conceptual (esquemas de estrutura) pelo esquema físico.

⇒ A base de dados conceptual é tida como incluindo todos os dados da organização.

Conceptual level ⇒ The community view of the database. This level describes *what* data is stored in the database and the relationships among the data.

Este nível contém a estrutura lógica de toda a base de dados. É uma visão global dos requisitos de dados da organização que é independente de qualquer restrição de armazenamento. O nível conceptual representa:

- todas as entidades, e seus atributos e relacionamentos;
- as restrições dos dados;
- informação semântica sobre os dados;
- informações de segurança e integridade.

O nível conceptual dá suporte às visões externas, na medida em que quaisquer dados disponíveis para qualquer utilizador devem estar presentes no nível conceptual, ou ser deriváveis a partir deste. No entanto, este nível não deve conter pormenores dependentes do armazenamento. Por exemplo, a descrição de uma entidade deve conter apenas os tipos de dados dos atributos (por exemplo, *integer*, *real*, ou *char*), mas não quaisquer considerações de armazenamento, como o número de bytes ocupados.

- Nível de visualização (“views” ou nível externo)

Uma “view” (subesquema) é uma porção da base de dados conceptual ou uma abstração de parte da base de dados conceptual.

Pode ser apenas uma pequena base de dados ao mesmo nível de abstração que a base de dados conceptual.

Pode estar a um nível de abstração mais elevado. Ou seja, os dados de uma “view” podem ser construídos a partir da base de dados conceptual, mas não estarem presentes na base de dados (*exemplo: - idade*).

External level \Rightarrow The users’ view of the database. This level describes that part of the database that is relevant to each user.

O nível externo consiste em várias visualizações externas sobre a base de dados. Cada utilizador tem uma vista do “mundo real” representado num formato familiar para esse utilizador. A vista externa inclui apenas as entidades, os atributos e os relacionamentos do “mundo real” em que o utilizador está interessado. Outros elementos (entidades, atributos e relacionamentos) podem estar presentes na base de dados, mas o utilizador não tem conhecimento deles.

Para além disso, diferentes vistas podem ter diferentes representações dos mesmos dados. Por exemplo, um utilizador pode visualizar datas no formato dia-mês-ano enquanto outro pode mostrar as datas como ano-mês-dia. Algumas vistas podem incluir dados derivados ou calculados: dados não armazenados na base de dados, mas criados quando necessários.

1.1.2.2 Independência de dados

Como resultado destes três níveis de abstração vamos ter dois níveis de independência de dados

Independência física de dados

Devido a questões de otimização do desempenho ou de segurança, é possível alterar aspetos relativos à implementação física da base de dados, sem que se altere o seu esquema conceptual, isto é, manter os dados e as associações entre eles inalterado.

Qualquer alteração no modelo físico não implicará alterações ou ajustamentos no modelo conceptual.

Exemplos:

- Alteração das estruturas de armazenamento (ficheiros)
- Criação de índices para otimizar o acesso aos dados
-

Independência lógica de dados

Durante o período de vida da base de dados pode ser necessário alterar o modelo conceptual, por exemplo, adicionando atributos a entidades já existentes ou criando entidades.

Muitas alterações podem ser feitas sem afetar vistas (“*views*”) já existentes.

Exemplo: Necessidade de registo de um novo atributo de uma entidade.
(Registo do “BI” de todas as “Pessoas”)

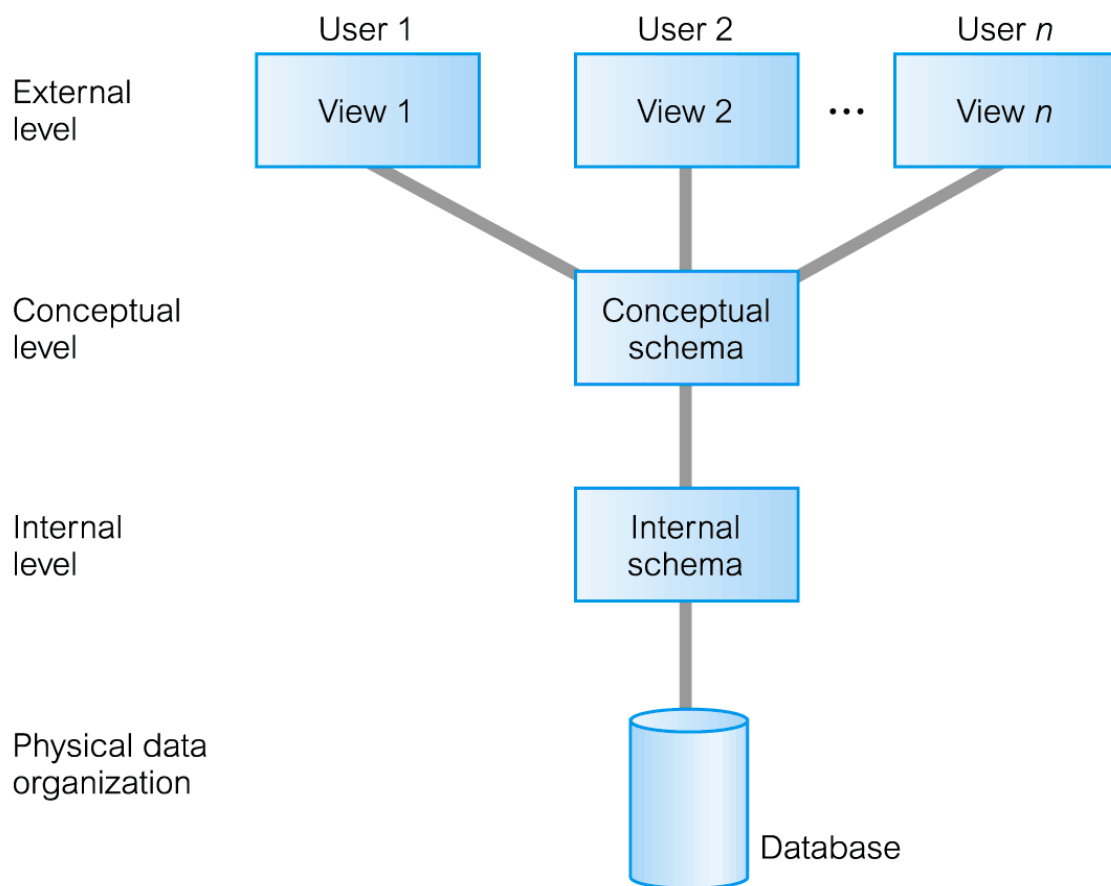
Eliminação de informação?

1.1.2.3 Arquitetura ANSI/SPARC

Arquitetura ANSI/SPARC para um sistema de bases de dados
(Proposta em 1975)

ANSI - American National Standards Institute

SPARC - Standards Planning and Requirements Committee



Os três níveis da arquitetura ANSI/SPARC¹.

Considere a arquitetura ANSI-SPARC dum SGBD e os seus níveis de abstração (externo, conceptual e interno). Explique o que se entende por independência lógica dos dados e por independência física dos dados?

¹ (Connolly & Begg, 2015), pág. 85.

Resp.:

Independência lógica de dados:

- Invariância dos subesquemas externos face a alterações no esquema conceptual;
- Será possível, na generalidade dos casos, alterar o esquema conceptual sem ter de alterar também o esquema externo. Ou seja, alterações no nível conceptual não interferem, de forma obrigatória, com as “vistas” estabelecidas no nível externo (a menos que haja eliminação de componentes no esquema conceptual, caso em que algumas das “vistas” estabelecidas no esquema externo poderão ser afetadas).

Independência física de dados:

- Capacidade de alterar o esquema interno (por exemplo, substituição das estruturas de armazenamento ou organização dos ficheiros) sem ter de alterar o esquema conceptual. Ou seja, alterações no nível interno não se repercutem no nível conceptual.
 - Isola o utilizador das alterações no armazenamento físico dos dados.
-

1.2 Objetivos e Capacidades de um SGBD

Um SGBD é um sistema de gestão e armazenamento de dados, capaz de aceder eficientemente a grandes quantidades de dados.

Serve de intermediário entre o nível aplicacional e a base de dados, evitando a manipulação direta por parte de cada aplicação cliente.

É a única entidade responsável pela segurança, integridade e validade dos dados armazenados.

A maioria dos SGBD possui:

- Suporte para pelo menos um modelo de dados através do qual o utilizador possa visualizar os dados.

- Suporte para linguagens de alto nível que permitam ao utilizador definir a estrutura de dados e manipular os dados.
- Gestão de transações - Possibilitar o acesso à base de dados de vários utilizadores em simultâneo (acesso concorrente)
- Controlo de acesso – Capacidade para impedir o acesso aos dados a utilizadores não autorizados e verificar a validade dos dados
- Capacidade de recuperação de falhas sem perda de dados.

1.2.1 Linguagens da base de dados

Nas linguagens de programação mais comuns, as instruções de declaração e de execução fazem parte de um só conjunto, isto é, estão englobadas numa mesma linguagem.

Em sistemas de bases de dados as duas funções estão separadas em duas linguagens específicas:

- Uma linguagem para definir a base de dados

Data Definition Language (DDL)

Utilizada para definir a estrutura da base de dados e da informação que deve armazenar.

Constitui uma notação para descrever a estrutura da informação.

- Uma linguagem de interrogação da Base de Dados
("query language")

Data Manipulation Language (DML)

É a linguagem disponibilizada para obter, armazenar, alterar ou eliminar informação da base de dados.

As instruções pertencentes a esta linguagem podem ser:

- Executadas de forma autónoma permitindo aos utilizadores finais usar diretamente a Base de Dados sem que programas de aplicação tenham de ser escritos

ou

- Embutidas em linguagens hospedeiras (C, Pascal, Visual Basic, Fortran, Java, ...).

Neste caso, um pré-compilador traduz as instruções DML para chamadas de sub-rotinas com os parâmetros correspondentes.

1.2.2 Transações

Numa transação, ou todas as instruções acabam ou nenhuma produz efeitos sobre a base de dados. Não há operações que sejam parcialmente completadas.

Exemplo típico: transferência de dinheiro entre duas entidades “A” e “B”.

Suponha-se que o cliente “A” efetuou uma compra de 1000 euros à empresa “B”.

É necessário debitar este valor na conta de “A” e creditar o mesmo valor na conta de “B”.

A transferência é composta por duas operações:

- Retirar 1000€ à conta de “A”
- Acrescentar 1000€ à conta de “B”.

Se uma situação excecional interrompe a transferência e a quantia não é creditada na conta de “B” a base de dados ficará inconsistente.

Uma transação é um conjunto de operações perfeitamente delimitado em que garantidamente são executadas todas as instruções ou então nenhuma produzirá efeitos sobre a base de dados.

Begin Transaction

Operação 1

Operação 2

...

Operação *n*

End Transaction

No exemplo da transferência as duas operações devem ser agrupadas numa transação.

1.2.3 Propriedades ACID

Uma transação deve exibir quatro características fundamentais:

(Propriedades ACID – Atomicity, Consistency, Isolation, Durability)

Atomicidade (Atomicity)

O conjunto de instruções que compõem a transação é indivisível, no sentido em que todas elas são executadas, ou então nenhuma produzirá efeitos.

Sempre que todas sejam executadas sem nenhuma situação excecional diz-se que foi executado o *COMMIT* da transação.

Na ocorrência de alguma situação excecional que impossibilite a sua completa execução, deve ser anulado o efeito de todas as instruções que compõem a transação e que já foram executadas (*ROLLBACK*).

Consistência (*Consistency*)

Uma transação deve, após a sua completa execução, deixar a base de dados num estado consistente.

Pode acontecer que durante a execução da transação a consistência não se verifique, no entanto após a execução de todas as suas operações, a consistência deve estar assegurada.

É ao utilizador que cabe a responsabilidade de assegurar a consistência de cada transação. O SGBS assume que a consistência de cada transação.

Isolamento (*Isolation*)

Apesar de ser possível a execução paralela e simultânea de diferentes transações, o sistema deve dar a ilusão de que cada uma delas é a única a executar, estando por isso aparentemente *isolada*.

Sempre que existam várias transações a aceder aos mesmos dados, o sistema deve evitar que existam interferências mútuas, e garantir que o estado final da base de dados é equivalente a (alguma) execução série das transações envolvidas.

Persistência (*Durability*)

Deve ser assegurado que após a execução bem-sucedida de uma transação (*COMMIT*), os seus efeitos são persistentes (não voláteis) na base de dados.

Qualquer transação futura deve operar sobre o novo conjunto de dados. Qualquer eventual falha não deve anular as alterações entretanto produzidas.

Os efeitos de cada transação podem apenas ser desfeitos ou alterados por outras transações.

1.2.4 Controlo de acesso

Segurança

É um dos requisitos básicos exigidos a um SGBD. Consiste em proteger os dados armazenados dos acessos não autorizados e garantir que todas as operações executadas sobre a base de dados o são por utilizadores (aplicações) devidamente credenciados.

Integridade

“Por definição, uma base de dados está num estado de integridade se todos os dados que contém são válidos, isto é, não contradizem a realidade que estão a representar nem se contradizem entre si.

Ao contrário das medidas de segurança, que se preocupam, fundamentalmente, em proteger a base de dados de acessos não

autorizados, a manutenção da integridade pressupõe proteger a base de dados de acessos menos válidos por parte de utilizadores autorizados, impedindo-os de executar operações que ponham em risco a correção dos dados armazenados.”².

Apenas as operações de atualização podem pôr em causa a base de dados. O SGBD permite definir regras que garantem a integridade após cada processo de alteração de dados.

Deve ser possível definir restrições de integridade do tipo:

- um item de dados pertencer a um conjunto de valores
- formato (natureza, comprimento)
- coerência com outros dados
- ...

A identificação das restrições de integridade é feita na fase de modelação de dados, sendo parte integrante do modelo conceptual.

1.2.5 Tolerância a Falhas

Devido à potencial importância dos dados armazenados numa base de dados, é essencial a implementação de mecanismos de tolerância a falhas (*hardware / software*), que garantam a reposição da informação para um estado anterior válido.

² (Pereira, 1990), pág. 48.

Para tal são utilizados basicamente dois mecanismos:

- Implementação de cópias de segurança com estados válidos da base de dados (*Backups*)
- Registos de atividade (*Logging*). Registo de todas as operações efetuadas sobre a base de dados a partir do último ponto de cópia

Se ocorre uma anomalia, o procedimento de recuperação permite a partir da última cópia e do registo de atividades, restaurar a base de dados para um estado consistente e detetar a origem da anomalia.

1.2.6 Arquitetura cliente-servidor

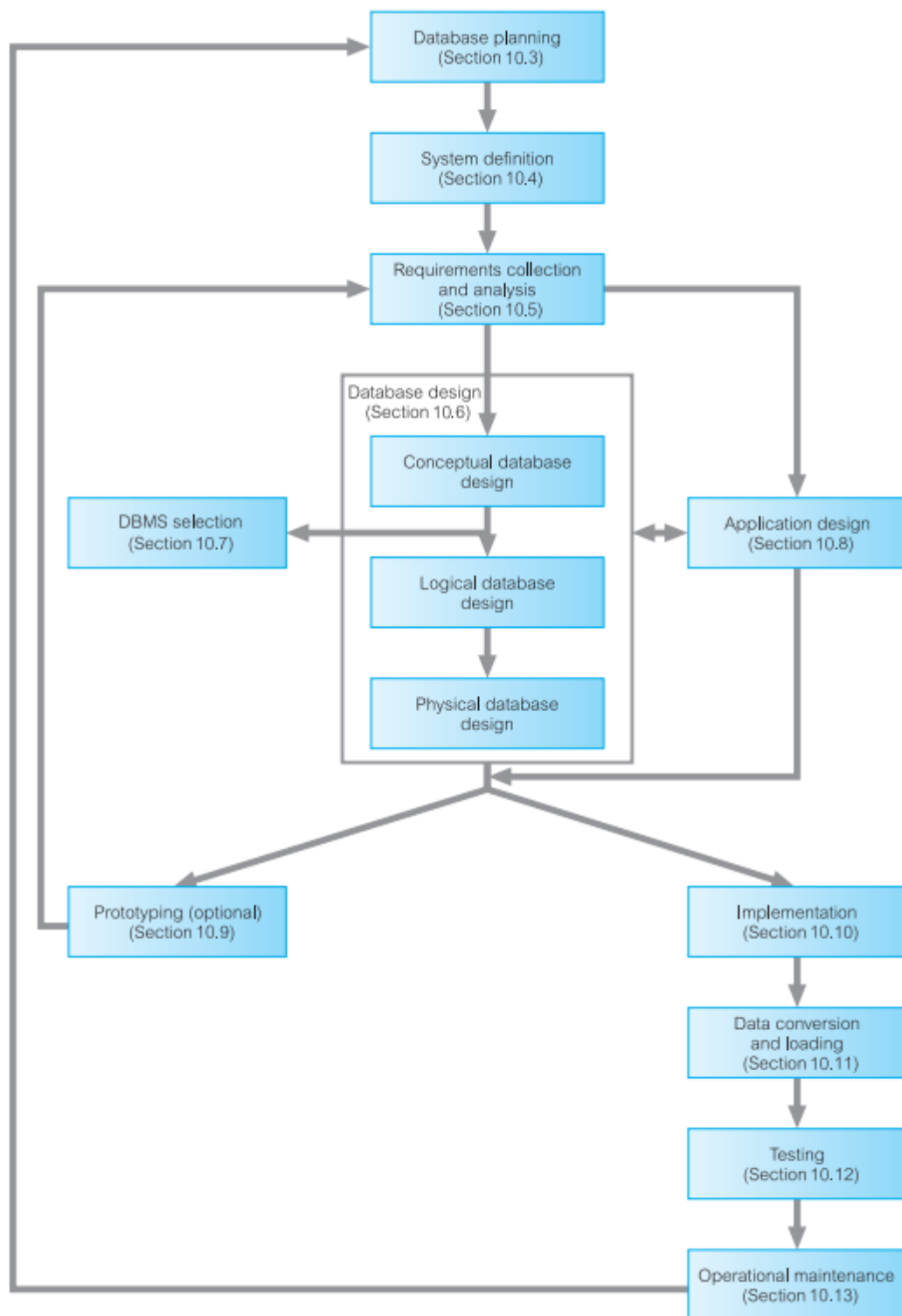
Tipicamente as aplicações de bases de dado seguem um modelo Cliente-servidor:

- Um servidor contém os dados e executa o SGBD
- Os Clientes ligados por uma rede de comunicações executam os programas de aplicação, que fazem a interface com o utilizador e o acesso remoto à base de dados.

Numa base de dados distribuída, esta surge ao utilizador como uma única base de dados, sendo na realidade constituída por diversas bases de dados (i. é, diversos SGBD) distribuídos por diferentes computadores.

1.3 Processo de desenvolvimento de sistemas de bases de dados³

1.3.1 Etapas do desenvolvimento de sistemas de bases de dados



³ Extraído de (Connolly & Begg, 2015), pág. 348.

Information system	The resources that enable the collection, management, control, and dissemination of information throughout an organization.
Database planning	The management activities that allow the stages of the database system development lifecycle to be realized as efficiently and effectively as possible.
System definition	Describes the scope and boundaries of the database system and the major user views.
User view	Defines what is required of a database system from the perspective of a particular job role (such as Manager or Supervisor) or enterprise application area (such as marketing, personnel, or stock control).
Requirements collection and analysis	The process of collecting and analyzing information about the part of the organization that is to be supported by the database system and using this information to identify the requirements for the new system.
Centralized approach	Requirements for each user view are merged into a single set of requirements for the new database system. A data model representing all user views is created during the database design stage.
View integration approach	Requirements for each user view remain as separate lists. Data models representing each user view are created and then merged later during the database design stage.
Database design	The process of creating a design that will support the enterprise's mission statement and mission objectives for the required database system.
Conceptual database design	The process of constructing a model of the data used in an enterprise, independent of <i>all</i> physical considerations.
Logical database design	The process of constructing a model of the data used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations.
Physical database design	The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.
DBMS selection	The selection of an appropriate DBMS to support the database system.
Application design	The design of the user interface and the application programs that use and process the database.

Transaction	An action, or series of actions, carried out by a single user or application program, that accesses or changes the content of the database.
Prototyping	Building a working model of a database system.
Implementation	The physical realization of the database and application design.
Data conversion and loading	Transferring any existing data into the new database and converting any existing applications to run on the new database.
Testing	The process of running the database system with the intent of finding errors.
Operational maintenance	The process of monitoring and maintaining the database system following installation.

1.3.2 Atividades principais do desenvolvimento

STAGE	MAIN ACTIVITIES
<i>Database planning</i>	Planning how the stages of the lifecycle can be realized most efficiently and effectively.
<i>System definition</i>	Specifying the scope and boundaries of the database system, including the major user views, its users, and application areas.
<i>Requirements collection and analysis</i>	Collection and analysis of the requirements for the new database system.
<i>Database design</i>	Conceptual, logical, and physical design of the database.
<i>DBMS selection</i>	Selecting a suitable DBMS for the database system.
<i>Application design</i>	Designing the user interface and the application programs that use and process the database.
<i>Prototyping (optional)</i>	Building a working model of the database system, which allows the designers or users to visualize and evaluate how the final system will look and function.

<i>Implementation</i>	Creating the physical database definitions and the application programs.
<i>Data conversion and loading</i>	Loading data from the old system to the new system and, where possible, converting any existing applications to run on the new database.
<i>Testing</i>	Database system is tested for errors and validated against the requirements specified by the users.
<i>Operational maintenance</i>	Database system is fully implemented. The system is continuously monitored and maintained. When necessary, new requirements are incorporated into the database system through the preceding stages of the lifecycle.

1.3.3 Critérios de produção de um modelo de dados ótimo

<i>Structural validity</i>	Consistency with the way the enterprise defines and organizes information.
<i>Simplicity</i>	Ease of understanding by IS professionals and nontechnical users.
<i>Expressibility</i>	Ability to distinguish between different data relationship between data and constraints.
<i>Nonredundancy</i>	Exclusion of extraneous information; in particular, the representation of any one piece of information exactly once
<i>Shareability</i>	Not specific to any particular application or technology and thereby usable by many.
<i>Extensibility</i>	Apply to evolve to support new requirements with minimal effect on existing users.
<i>Integrity</i>	Consistency with the way the enterprise uses and manages information.
<i>Diagrammatic representation</i>	Ability to represent a model using an easily understood diagrammatic notation.

2 Modelos de Dados

2.1 Introdução

Um **modelo de dados** é a **coleção de**, pelo menos, 3 componentes:

- 1) Um conjunto de tipos de estruturas de dados

Define o tipo de dados e como se interrelacionam

- 2) Um conjunto de operadores

Operações que permitem manipular as estruturas de dados definidas.

- 3) Um conjunto de regras de integridade

Regras que definem que dados são válidos

1ª Geração de SGBD's (*meados dos anos 60*):

Modelo Hierárquico

Modelo em Rede ou *Codasyl*

2ª Geração

Modelo Relacional (*proposto em 1970 por E. F. Codd*)

3ª Geração

Modelo *Object - Oriented*

...

Suponhamos a **Base de Dados Exemplo**:

Obra

- N_obra
- Nome_obra
- Data_de_inicio
- Data_de_fim
- Orçamento

Fornecedor

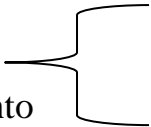
- N_fornecedor
- Nome_fornecedor
- Morada

Material

- N_material
- Nome_material
- Unidade_medida

Fornecimento

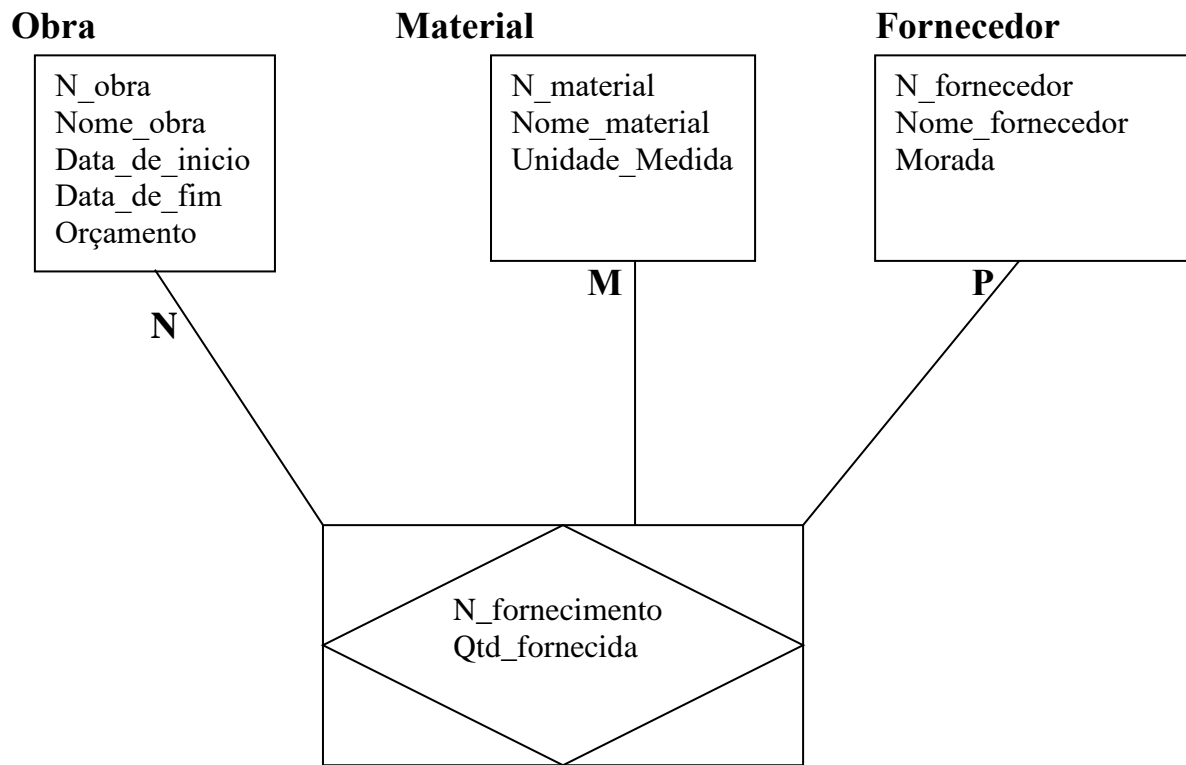
N_fornecimento



- N_obra
- N_material
- N_fornecedor

Qtd_fornecida

Modelo Conceptual



- . Cada fornecedor pode fornecer vários materiais (M) para várias obras (N).
- . Cada material pode ser fornecido por vários fornecedores (P) para várias obras (N).
- . Cada obra pode receber vários materiais (M) de vários fornecedores (P).

Como realizar as operações que se seguem nos modelos Hierárquico, Rede e Relacional?

. Interrogações

I1: Quem forneceu o material *MI* para a obra *O1*?

I2: Que materiais forneceu o fornecedor *F2* e para que obras?

. Atualizações:

A1: Apagar todos os fornecimentos do fornecedor *F1* para a obra *O1*.

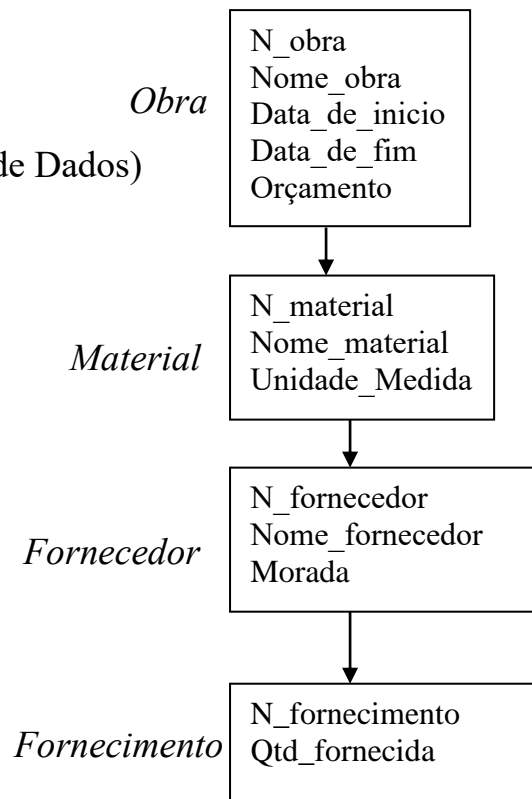
A2: Juntar à base de dados um novo fornecedor *F5*.

A3: Modificar a morada do fornecedor *F3* para “Covilhã”

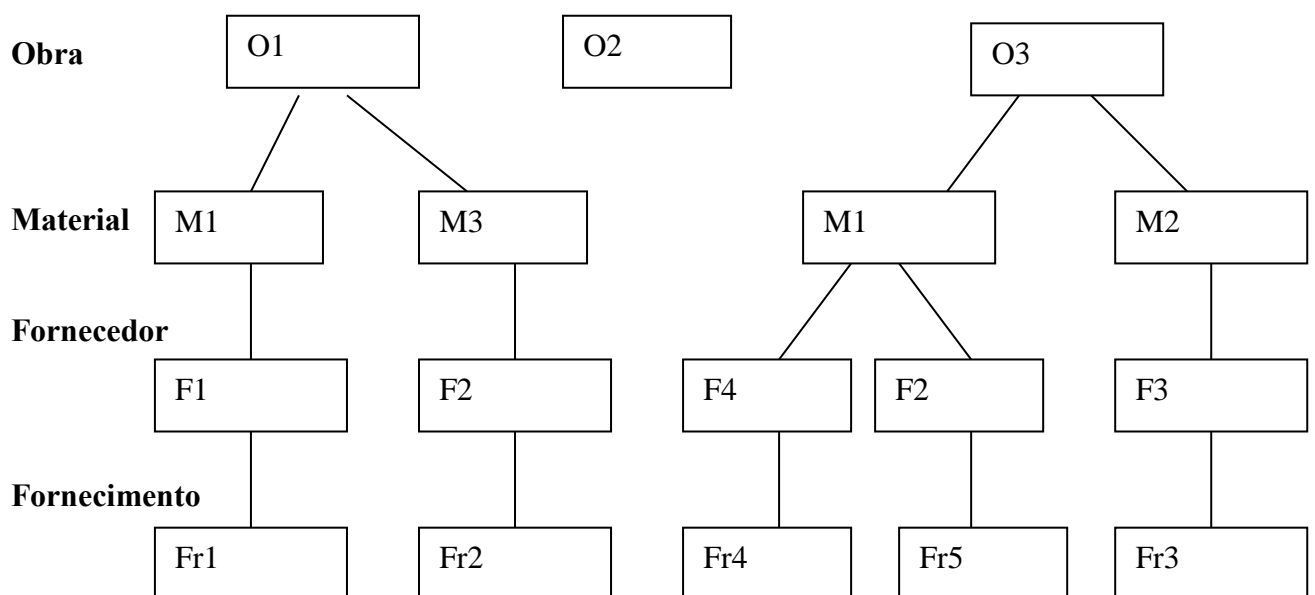
2.1.1 Modelo Hierárquico

Um esquema possível:

(Diagrama da Estrutura de Dados)



Conteúdo da Base de Dados num determinado instante:



Operação elementar de interrogação do modelo hierárquico:

Get [Next | Superior] <nome do registo 1>

[For This <nome do registo 2> **AND**

This <nome do registo 3> ...(*) **]**

[Where <condição lógica> **]**

(*) até ao nível anterior ao registo 1

[] significa opcional.

I1: Quem forneceu o material *MI* para a obra *OI*?

Get **Obra** Where **N_obra = O1**

Get **Material** For This **Obra**

Where **N_material = M1**

Get Next **Fornecedor**

For This **Obra** AND This **Material**

Do While **not end-of-fornecedor**

Print **N_fornecedor, Nome_fornecedor, Morada**

Get Next **Fornecedor**

For This **Obra** AND This **Material**

End Do

I2: *Que materiais forneceu o fornecedor F2 e para que obras?*

...

Operações elementares de atualização do modelo hierárquico:

. Insert Into <nome do registo>

<lista de valores>

[**Linking** < chave (!) do 1º nível = valor1 ,
chave do 2º nível = valor2, *]

Nas operações seguintes é necessário em primeiro lugar encontrar o registo (com *Get*) e só depois apagá-lo ou modificá-lo.

. Delete <nome do registo>

. Update <nome do registo>

Setting <lista de modificações>

(*) até ao nível anterior ao do registo

A1: Apagar todos os fornecimentos do fornecedor F1 para a obra O1.

A2: Juntar à base de dados um novo fornecedor F5.

A3: Modificar a morada do fornecedor F3 para “Covilhã”

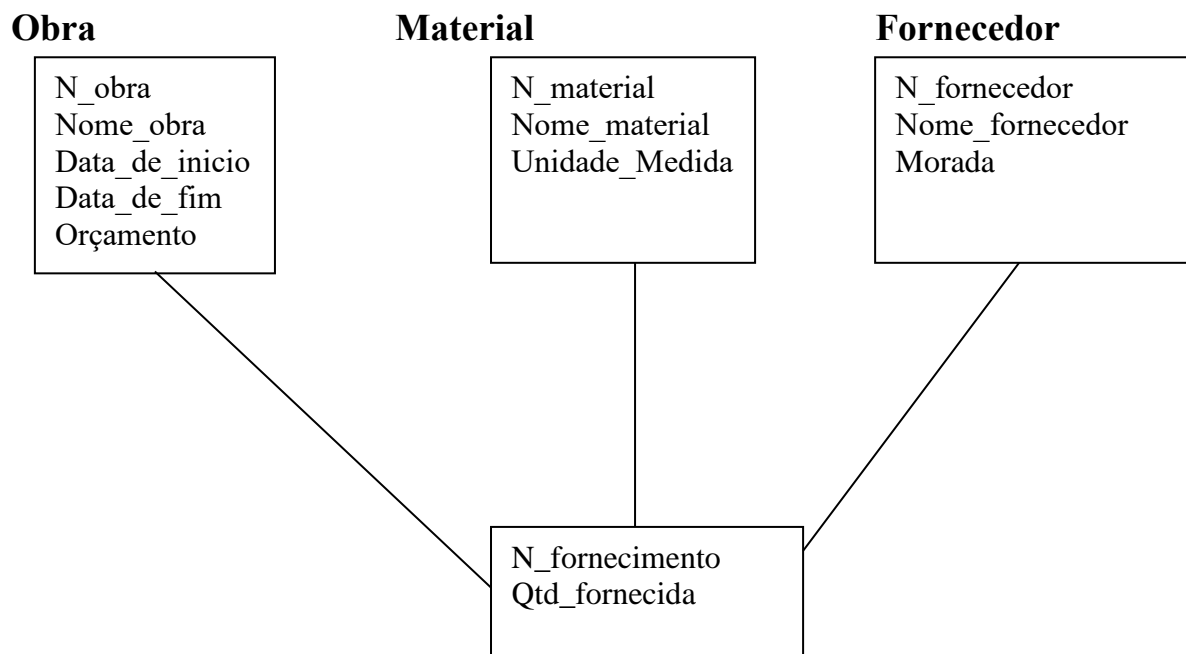
...

2.1.2 Modelo em Rede

*Definido pelo DBTG (Data Base Task Group) da CODASYL (CO~~n~~ference on **D**ata **S**ystems **L**anguages)*

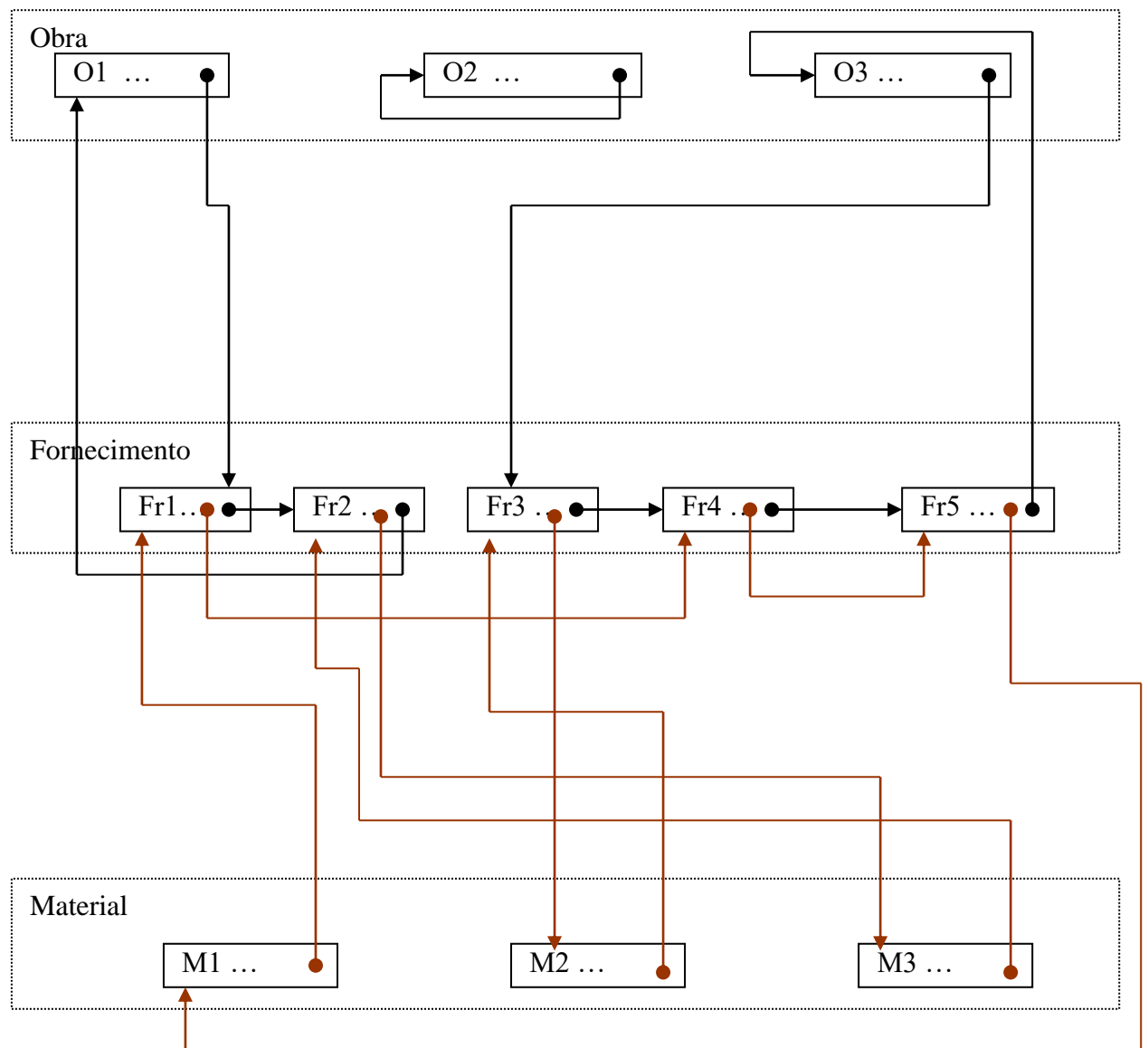
Esquema da base de dados:

(Diagrama da Estrutura de Dados)



- . Uma obra tem vários fornecimentos.
- . Um material tem vários fornecimentos
- . Um fornecedor faz vários fornecimentos.

Conteúdo da base de dados:



Completar para Fornecedor / Fornecimento ...

Operação elementar de interrogação do modelo em rede:

Get [Next | Superior] <nome do registo 1>

[For This <nome do registo 2> **]**

[Where <condição lógica> **]**

I1: Quem forneceu o material *M1* para a obra *O1*?

Get **Material**

Where **N_material = M1**

Get Next **Fornecimento**

For This **Material**

DO While not end-of-fornecimento

Get Superior **Obra**

For This **Fornecimento**

If **N_obra = o1** Then

Get Superior **Fornecedor**

For This **Fornecimento**

Print N_fornecedor, nome_fornecedor, morada

End IF

Get Next **Fornecimento**

For This **Material**

End Do

I2: Que materiais forneceu o fornecedor F2 e para que obras?

...

Operações elementares de atualização do modelo em rede:

. Insert Into <nome do registo>

<lista de valores>

[**Linking** < chave do registo superior 1 = valor1 , ...]

Nas operações seguintes é necessário em primeiro lugar encontrar o registo (com *Get*) e só depois apagá-lo ou modificá-lo.

. Delete <nome do registo>

. Update <nome do registo>

Setting <lista de modificações>

A1: Apagar todos os fornecimentos do fornecedor F1 para a obra O1.

A2: Juntar à base de dados um novo fornecedor F5.

A3: Modificar a morada do fornecedor F3 para “Covilhã”

...

Nota:

As linguagens de manipulação de dados dos modelos hierárquico e rede são ainda linguagens procedimentais – nestas linguagens o utilizador indica quais os registos a que quer aceder especificando como aceder a esses registos.

Numa linguagem não procedimental (como por exemplo o SQL) o utilizador indica os dados que pretende, mas não especifica a forma de obter esses dados.

2.1.3 Modelo Relacional

Dada uma coleção de conjuntos D_1, D_2, \dots, D_n (não necessariamente disjuntos), R é uma **relação** naqueles conjuntos se for constituída por um conjunto de n-uplos ordenados $\langle d_1, d_2, \dots, d_n \rangle$ tais que: $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$.

Domínios de R: D_1, D_2, \dots, D_n

Cardinalidade de R: número de n-uplos (tuplos) da relação

Grau de R: n

Base de Dados Relacional: conjunto de relações cujo conteúdo varia ao longo do tempo.

Esquema de Relação: definição de uma relação.

Esquema relacional: Definição de uma base de dados relacional. É uma coleção de esquemas de relação.

Exemplo (de um esquema relacional):

Obra (N_obra, Nome_Obra, Data_de_inicio, Data_de_fim, Orçamento)

Fornecedor (N_fornecedor, Nome_fornecedor, Morada)

Material (N_material, Nome_material, Unidade_de_medida)

Fornecimento (N_fornecimento, N_obra, N_fornecedor, N_material,
Qtd_fornecida)

Conteúdo da Base de Dados (num determinado instante de tempo):

Obra

N_obra	Nome_obra	Data_de_inicio	Data_de_fim	Orçamento
O1	Fábrica X	20-01-2004	30-07-2007	1 000 000
O2	Hospital Y	15-10-2003	20-12-2005	800 000
O3	Escola Z	20-01-2005	30-09-2005	50 000

- Qual é a cardinalidade de Obra?

- Qual é o grau de Obra?

Fornecedor

N_fornecedor	Nome_fornecedor	Morada
F1	Empresa A	Lisboa
F2	Empresa B	Portalegre
F3	Empresa C	Coimbra
F4	Empresa D	Lisboa

Material

N_material	Nome_material	Unidade_de_medida
M1	Cimento	Kg
M2	Ferro	Kg
M3	Areia	Kg

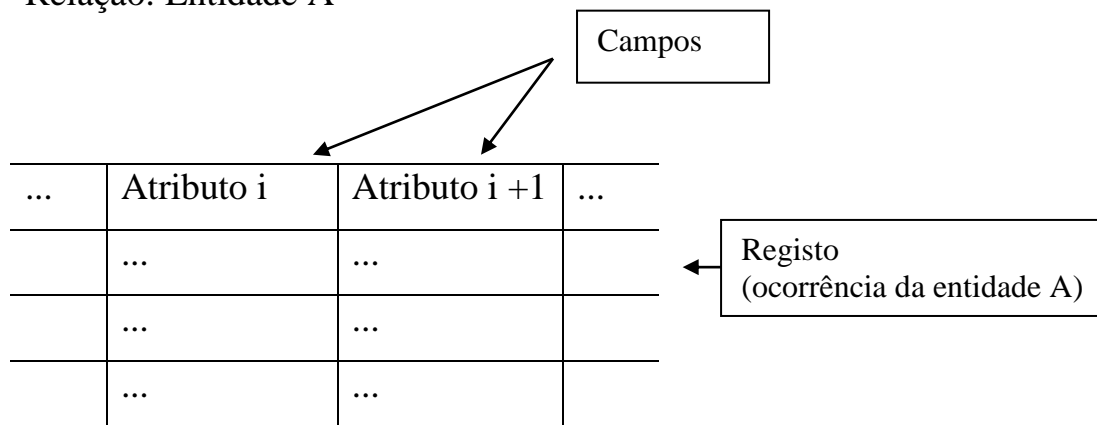
Fornecimento

N_fornecimento	N_obra	N_fornecedor	N_material	Qtd_fornecida
Fr1	O1	F1	M1	10 000
Fr2	O1	F2	M3	5 000
Fr3	O3	F3	M2	500
Fr4	O3	F4	M1	1 000
Fr5	O3	F2	M1	50 000

Para se definir o modelo de dados há que identificar as entidades que fazem parte do sistema que queremos representar, as suas características ou atributos e as associações entre as entidades.

Na prática, numa base de dados relacional os dados residem em tabelas (relações).

Relação: Entidade A



Terminologia

Formal	Implementação	Analogia
Relação	<i>Tabela</i>	<i>Ficheiro</i>
<i>Tuplo</i>	<i>Linha</i>	<i>Registo</i>
<i>Atributo</i>	<i>Coluna</i>	<i>Campo</i>

Pode definir-se relação como o produto cartesiano de todos os domínios de cada atributo.

- Cada relação contém um só tipo de registo
- Os campos não têm qualquer ordem
- Os registos não têm qualquer ordem
- Os registos têm um identificador único, constituído por um campo ou uma associação de campos, denominado chave primária.
- Em qualquer instante não há registos duplicados
- Para encontrar dados armazenados em qualquer localização da relação só é necessário nomear a relação e especificar a intersecção do campo e do registo.
- Um registo não pode ter campos de grupo. Um nome no interior de uma relação refere-se unicamente a um campo.

Operações

A interrogação de bases de dados baseadas nos modelos hierárquico e de rede (CODASYL) envolvia lidar com estruturas de baixo nível, o que implicava um elevado conhecimento técnico. As aplicações dependiam da forma de representação dos dados e de acesso aos mesmos. Qualquer alteração da implementação física dos dados obriga à alteração dos programas que os utilizam.

A álgebra relacional e o cálculo relacional, definidos por (Codd, 1970), são a base das linguagens relacionais.

*“Informalmente, podemos descrever a **álgebra relacional** como uma linguagem procedimental (alto nível): pode ser usada para dizer ao SGBD como construir uma nova relação a partir de uma ou mais relações na base de dados.*

*Ainda informalmente, podemos descrever o **cálculo relacional** como uma linguagem não-procedimental: pode ser usada para formular a definição de uma relação em termos de uma ou mais relações da base de dados.*

No entanto, a álgebra relacional e o cálculo relacional são formalmente equivalentes: para cada expressão na álgebra, há uma expressão equivalente no cálculo (e vice-versa). ”⁴

2.1.3.1 Álgebra Relacional

“A álgebra relacional é uma linguagem teórica que opera em uma ou mais relações para definir uma nova relação sem, contudo, alterar a relação original. Portanto, tanto os operandos como o resultado são relações, donde o output de uma operação pode ser o input de outra operação.”⁴

Consiste numa coleção de operações sobre relações:

Projeção
Restrição
Junção
...

⁴ (Connolly & Begg, 2015) pág. 167.

2.1.3.1.1 Projeção

Permite a redução de dados a valores únicos.

Permite ver o conjunto de valores de um dado atributo.

Exemplo:

1) *Projeção da relação Fornecedor sobre o atributo Morada*

Morada
Lisboa
Portalegre
Coimbra

2) *Projeção da relação Material, no atributo Unidade_de_Medida*

Unidade_de_medida
Kg

2.1.3.1.2 Restrição (ou seleção)

Permite seleccionar os “registos” que se pretendem manipular, de acordo com uma determinada condição.

Exemplo:

1) *Restrição da relação Fornecedor tal que Morada = “Lisboa”*

N_fornecedor	Nome_fornecedor	Morada
F1	Empresa A	Lisboa
F4	Empresa D	Lisboa

2) Restrição da relação *Obra* tal que $Data_de_fim > 30-10-2005$

N_obra	Nome_obra	Data_de_inicio	Data_de_fim	Orçamento
O1	Fábrica X	20-1-2004	30-7-2007	1 000 000
O2	Hospital Y	15-10-2003	20-12-2005	800 000

3) Restrição da relação *Fornecimento* tal que

$$N_Obra = O3 \wedge Qtd_fornecida > 5\,000$$

N_fornecimento	N_obra	N_fornecedor	N_material	Qtd_fornecida
Fr5	O3	F2	M1	50 000

2.1.3.1.3 Junção natural (Natural Join)

Junção natural das relações *A* e *B* sobre os atributos *X* e *Y*.

(*X* e *Y* têm de ter o mesmo domínio).

. Selecciona cada registo de uma relação, associa-o com o registo correspondente da outra relação e apresenta-os como se fizessem parte de um único registo.

Exemplo:

Supondo as seguintes relações:

Fornecedor

NF	Nome	Cidade
F1	João	Lisboa
F2	Maria	Porto
F3	Mário	Coimbra

Fornecimento

N_Fr	NF	NM	Qtd
Fr1	F1	M1	300
Fr2	F1	M2	200
Fr3	F1	M3	400
Fr4	F2	M1	300
Fr5	F2	M2	400
Fr6	F3	M2	200
Fr7	F4	M1	500

Junção das relações Fornecedor e Fornecimento sobre NF (da relação Fornecedor) e NF (da relação Fornecimento)

NF	Nome	Cidade	N_fr	NM	Qtd
F1	João	Lisboa	Fr1	M1	300
F1	João	Lisboa	Fr2	M2	200
F1	João	Lisboa	Fr3	M3	400
F2	Maria	Porto	Fr4	M1	300
F2	Maria	Porto	Fr5	M2	400
F3	Mário	Coimbra	Fr6	M2	200

Vamos voltar às questões colocadas para a nossa 1ª base de dados exemplo:

II: Quem forneceu o material M1 para a obra O1?

a) Restrição da relação Fornecimento tal que

$$N_material = M1 \wedge N_obra = O1$$

(Fornecimento) Temp1

N_fornecimento	N_obra	N_fornecedor	N_material	Qtd_fornecida
Fr1	O1	F1	M1	10 000

b) Junção das relações Temp1 e Fornecedor sobre N_fornecedor

Temp2

N_fornecimento	N_obra	N_fornecedor	N_material	Qtd_fornecida	Nome_fornecedor	Morada
Fr1	O1	F1	M1	10 000	Empresa A	Lisboa

c) Projeção da relação *Temp2* no atributo *Nome_fornecedor*

Resposta:

Nome_fornecedor
Empresa A

I2: Que materiais (nome) forneceu o fornecedor F2 e para que obras (nomes)?

a) Restrição da relação *Fornecimento* tal que $N_{fornecedor} = F2$:

Temp1

N_fornecimento	N_obra	N_fornecedor	N_material	Qtd_fornecida
Fr2	O1	F2	M3	5 000
Fr5	O3	F2	M1	50 000

b) Junção das relações *Material* e *Temp1* sobre $N_{material}$

Temp2

N_fornecimento	N_obra	N_fornecedor	N_material	Qtd_fornecida	Nome_material	Unidade..
Fr2	O1	F2	M3	5 000	Areia	Kg
Fr5	O3	F2	M1	50 000	Cimento	Kg

c) Junção das relações *Obra* e *Temp2* sobre *N_obra*

Temp 3

N_for./o	N_ob	N_for	N_mat.	Qtd_forn.	Nome_mat.	Unid.	Nome_obra	Data_de
Fr2	O1	F2	M3	5 000	Areia	Kg	Fabrica X	...
Fr5	O3	F2	M1	50 000	Cimento	Kg	Escola Z	...

d) Projeção da relação *Temp3* nos atributos *Nome_material* e *Nome_obra*

Resposta:

Nome_material	Nome_obra
Areia	Fábrica X
Cimento	Escola Z

2.1.3.2 Chaves

- **Chave candidata** de uma relação: atributo ou conjunto de atributos que permitem identificar de forma inequívoca qualquer tuplo dessa relação. O conjunto não pode ser reduzido sem perder essa qualidade.

De entre as possíveis chaves candidatas é escolhida uma que será declarada como chave Primária

- A **Chave Primária** terá de ser:
 - Unívoca: o atributo (ou atributos) da chave primária têm um valor único para qualquer tuplo da relação.

- Não nula: Não pode haver tuplos da relação que tenham o atributo (ou atributos) da chave primária nulos (sem qualquer valor).
- Não redundante: Se algum dos atributos que a constituem for retirado os restantes deixam de identificar univocamente o tuplo.

Nome	B.I	N_contribuinte	N_eleitor	Freguesia	Concelho
Maria	1234567	123456722	2222	S. Pedro	Covilhã
Manuel	3377229	234156233	3333	Conceição	Covilhã
Paulo	2233337	233333567	3456	S. Maria	Covilhã
Paula	2876909	222333333	6782	S. Tiago	Covilhã

Exemplo

Chaves candidatas: {B.I.}, {N_Contribuinte},
{N_Eleitor, Freguesia, Concelho}

Chave Primária: ?

- É **superchave** de uma relação, qualquer subconjunto de atributos que identifique univocamente qualquer tuplo da relação.
 - No limite o conjunto de todos os atributos da relação é uma superchave.

Exemplos:

{BI}, {BI, Nome}, {N_Eleitor, Freguesia, Concelho}, {N_Eleitor, BI},
{N_Eleitor, BI, Nome}, {Nome, BI, N_Contribuinte, N_Eleitor, Freguesia, Concelho}, ...

- **Chave Estrangeira**: Subconjunto de atributos que constituem a chave primária de uma outra relação permitindo estabelecer a associação entre tuplos de diferentes relações.

Exemplo:

(ver base de dados exemplo)

Fornecimento

N_fornecimento	N_obra	N_fornecedor	N_material	Qtd_fornecida
Fr1	O1	F1	M1	10 000
Fr2	O1	F2	M3	5 000
Fr3	O3	F3	M2	500
Fr4	O3	F4	M1	1 000
Fr5	O3	F2	M1	50 000

- *N_fornecimento* é chave primária da relação *Fornecimento*;
- *N_obra* é chave estrangeira da relação *Fornecimento* porque é chave primária na relação *Obra*;
- *N_fornecedor* é chave estrangeira da relação *Fornecimento* porque é chave primária na relação *Fornecedor*;
- *N_material* é chave estrangeira da relação *Fornecimento* porque é chave primária na relação *Material*.

2.2 Modelo Relacional

2.2.1 Estrutura de Dados Relacional

2.2.1.1 Modelo Conceptual de Dados

*Um **modelo conceptual** de dados é a representação de um conjunto de objetos e das suas associações*

Como qualquer representação é o resultado de um processo de abstração.

. Durante esse processo de abstração, objetos relevantes, associações entre eles e características (atributos) de objetos e associações são selecionadas.

. A relevância de um objeto, de uma associação ou de um atributo é determinada pelos objetivos do modelo.

. Atributos de objetos e correspondentes associações têm valores específicos que pertencem a conjuntos denominados domínios.

. Um valor de um dado atributo pode variar ao longo do tempo, mas pertencendo sempre ao domínio desse atributo.

O modelo relacional baseia-se no pressuposto de que os dados (que obedecem a certas restrições) podem ser tratados da mesma forma que as relações matemáticas.

2.2.1.2 Entidades, Atributos e Domínios

Objetos e respectivas associações são chamados ENTIDADES.

Um conjunto E de entidades do mesmo tipo é caracterizado por um conjunto de ATRIBUTOS A_1, A_2, \dots, A_n e denotado por

$$E(A_1, A_2, \dots, A_n)$$

onde

$A_i : E \rightarrow D_i$ é uma função cujo contradomínio D_i é denominado DOMÍNIO do atributo A_i .

Dado $e \in E$, $A_i(e) \in D_i$ é denominado o valor do atributo A_i da entidade e .

Exemplo de um tipo de entidade

Seja o tipo de entidade *Pessoa* cujos atributos relevantes são:

Número de segurança social
Primeiro nome
Último nome
Idade

Pessoa (NSS, P_nome, U_nome, Idade)

Onde os domínios dos atributos são:

NSS – o conjunto, S , dos números de segurança social

P_nome – o conjunto, A , de sequências finitas de letras

U_nome – o conjunto, A , de sequências finitas de letras

Idade – o conjunto, N , dos inteiros positivos <150 !

2.2.1.3 Representação de entidades por tuplos

Dado um tipo de entidade

$$E (A_1, A_2, \dots, A_n)$$

o conjunto de funções

$$A_1: E \rightarrow D_1, \quad A_2: E \rightarrow D_2, \dots, A_n: E \rightarrow D_n$$

determina uma única função:

$$(A_1, A_2, \dots, A_n) : E \rightarrow D_1 \times D_2 \times \dots \times D_n \quad (i)$$

onde

$D_1 \times D_2 \times \dots \times D_n$, denota o produto cartesiano dos conjuntos D_1, D_2, \dots, D_n

(isto é, o conjunto de todos os n -uplos (d_1, d_2, \dots, d_n) onde $d_i \in D_i$ para $i = 1, 2, \dots, n$)

A função (A_1, A_2, \dots, A_n) é definida como:

$$(A_1, A_2, \dots, A_n) (e) = (A_1(e), A_2(e), \dots, A_n(e)) \quad (ii)$$

onde para quaisquer duas entidades e_1 e e_2 do tipo E se verifique que

$$(A_1, A_2, \dots, A_n) (e_1) \neq (A_1, A_2, \dots, A_n) (e_2) \quad (iii)$$

A condição (iii) verifica-se se existir um $j = 1, 2, \dots, n$ para o qual

$$A_j (e_1) \neq A_j (e_2)$$

Diferentes entidades são representadas por tuplos diferentes.

Dois tuplos são diferentes se têm valores diferentes em pelo menos um atributo.

Exemplo da representação de um tipo de entidade por um conjunto de tuplos

A notação

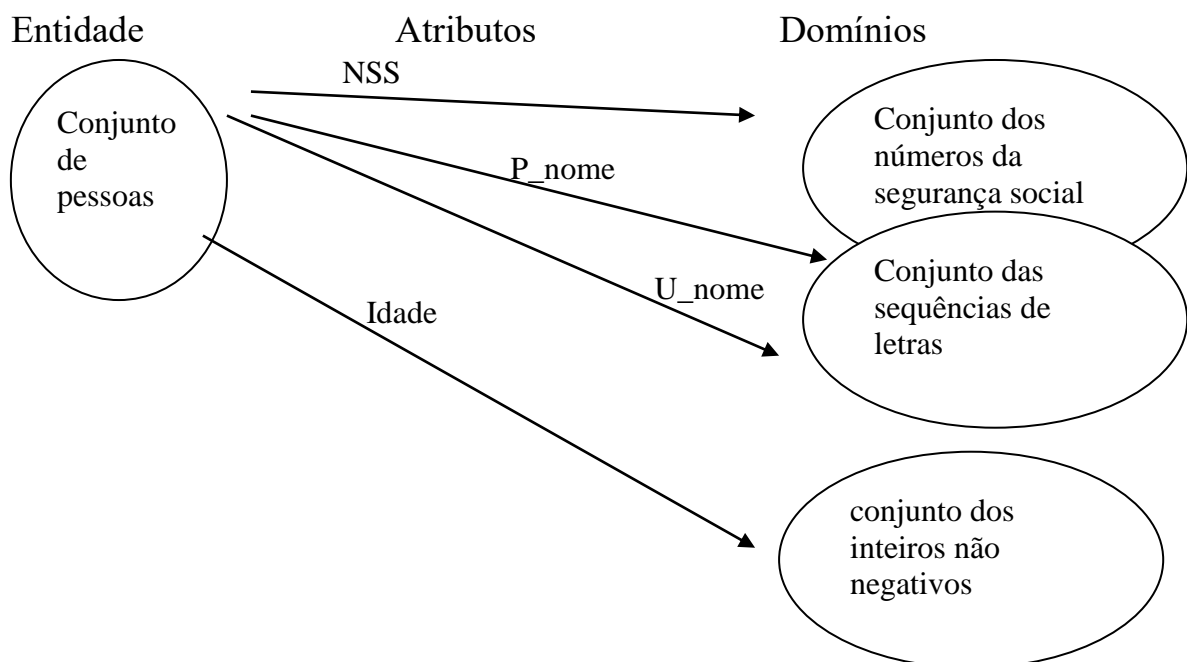
Pessoa (*NSS*, *P_nome*, *U_nome*, *Idade*) é interpretada como a função

$$(NSS, P_nome, U_nome, Idade): Pessoa \rightarrow S \times A \times A \times N$$

Esta função determina para cada pessoa p um 4-uplo (*nº de segurança social*, *primeiro nome*, *último nome*, *idade*) que representa essa pessoa.

Diferentes pessoas p_1 e p_2 determinam diferentes tuplos.

Mesmo que tenham os mesmos primeiro e último nomes, o *NSS* é seguramente diferente.



Num dado instante, um conjunto de pessoas pode ser representado pela seguinte tabela:

NSS	P_nome	U_nome	Idade
941	Pedro	Silva	31
385	Mário	Sousa	24
102	Joana	Ferreira	64
243	Maria	Andrade	52
860	João	Almeida	24
543	Alice	Fonseca	45

2.2.1.4 Relação

R é uma relação nos conjuntos D_1, D_2, \dots, D_n se e só se

$$R \subseteq D_1 \times D_2 \times \dots \times D_n$$

A estrutura da relação R é descrita pela notação $R(A_1, A_2, \dots, A_n)$ onde

$A_i: R \rightarrow D_i$ é um atributo de R e D_i o seu domínio para $i = 1, 2, \dots, n$.

2.2.1.5 Base de dados relacional e esquema relacional

Uma base de dados relacional é uma coleção de relações cujo conteúdo varia ao longo do tempo.

Um esquema relacional é a descrição da estrutura das relações numa base de dados relacional.

Exemplo de um esquema relacional

Departamento (Dep#, Nome, Local)

Empregado (Emp#, Nome, Categoria, Dep#)

Projeto (Proj#, Designação, Fundos)

Atribuição (Emp#, Proj#, Função)

O esquema descreve 4 tipos de entidades:

Departamento, Empregado, Projeto, e Atribuição de empregados a projetos.

Atributos de entidades do tipo *Departamento* são:

Dep# – número de departamento

Nome – nome do departamento

Local – localização do departamento

Atributos de entidades do tipo *Empregado* são:

Emp# – nº de segurança social do empregado

Nome – nome do empregado

Categoria – Categoria do empregado

Dep# – número do departamento a que pertence o empregado

Atributos de entidades do tipo *Projeto* são:

Proj# – Código do projeto

Designacao – designação do projeto

Fundos – fundos atribuídos ao projeto

Atributos de entidades do tipo *Atribuição*

Emp# – número de segurança social do empregado

Proj# – código do projeto

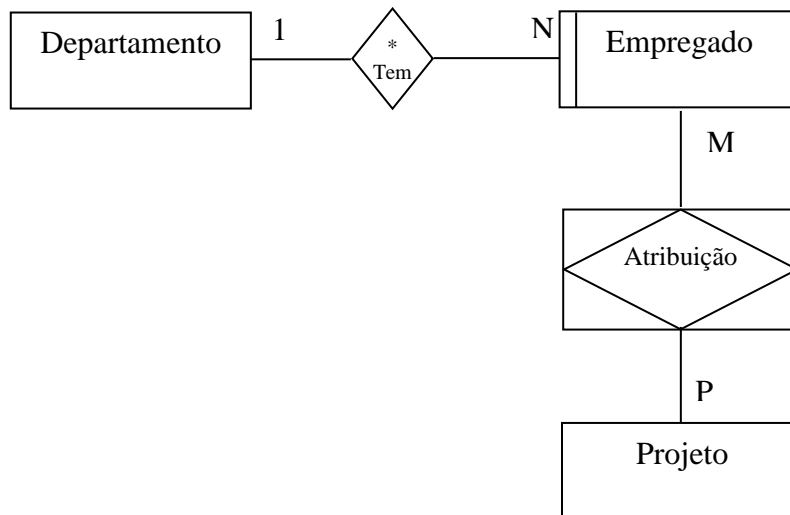
Funcao – função que o empregado desempenha no projeto

São assumidas as seguintes restrições:

- um empregado pertence a um único departamento

- um empregado pode ser designado para vários projetos e um projeto tem vários empregados atribuídos

Representação do Diagrama Entidade-Associação:



Como obteríamos a resposta da interrogação seguinte:

Obter os nomes dos empregados com categoria “Programador” e pertencentes a departamentos localizados em “Lisboa”?

2.2.2 Álgebra Relacional

Um modelo por si próprio não pode realizar qualquer unidade de trabalho útil. É apenas uma representação da realidade.

Para realizar interrogações acerca das propriedades das entidades representadas no modelo precisamos de uma linguagem apropriada.

Existem várias linguagens eficientemente implementadas e amplamente aceites. Do ponto de vista conceptual todas tiveram origem numa linguagem formal denominada **Álgebra Relacional**.

Uma interrogação (*query*) em álgebra relacional consiste numa coleção de operadores sobre relações. Cada operador aceita um ou dois operandos (relações) e devolve como resultado uma relação.

Cada *query* descreve um procedimento passo-a-passo para calcular a resposta desejada, baseado na ordem em que os operadores são aplicados na *query*.

Operações usuais sobre conjuntos:

- União
- Intersecção
- Diferença
- Produto cartesiano

Outras operações:

- Projeção
- Restrição
- Junção
- Divisão

2.2.2.1 Projeção

Seja $R(X, Y)$ com $X = A_1, A_2, \dots, A_k$

$$Y = A_{k+1}, \dots, A_n$$

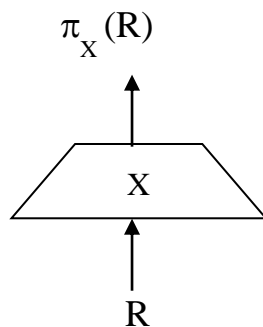
Projeção de R sobre os atributos X :

$$\pi_X(R) = \{x \mid \exists y : (x, y) \in R\}$$

Se a relação R é representada como uma tabela, a operação de projeção de R sobre o conjunto de atributos X é interpretada como a seleção das colunas de R que correspondem aos atributos de X e a eliminação das linhas duplicadas na tabela obtida.

	//////////	
	//////////	
	//////////	
	//////////	

Representação gráfica:



Exemplo:

Empregado (Emp#, Nome, Categoria, Dep#)

. Emp# é chave da relação empregado

Empregado

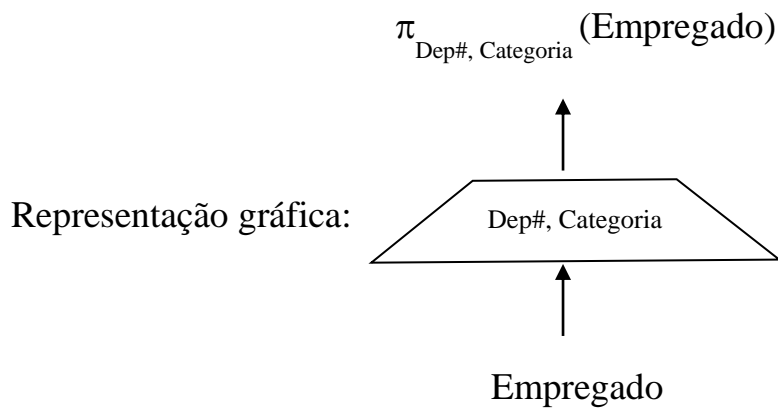
Emp#	Nome	Categoria	Dep#
e1	n1	c1	d1
e2	n2	c2	d2
e3	n3	c3	d1
e4	n4	c1	d2
e5	n5	c2	d3
e6	n6	c2	d3
e7	n7	c1	d1

Projeção da tabela *Empregado* sobre os atributos *Dep#* e *Categoria*,

$\pi_{\text{Dep\#, Categoria}}(\text{Empregado})$

dá origem à tabela:

Dep#	Categoria
d1	c1
d1	c3
d2	c1
d2	c2
d3	c2



2.2.2.2 Restrição (ou seleção)

Seja a relação $R(A_1, A_2, \dots, A_n)$ e p uma expressão lógica (predicado) definida sobre $D_1 \times D_2 \times \dots \times D_n$, com D_i domínio de A_i .

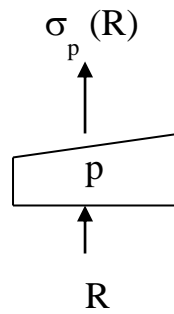
A restrição de R a respeito da condição (predicado) p ,

$$\sigma_p(R) = \{z \mid z \in R \wedge p(z) \text{ é verdadeiro}\}$$

Sendo R representada como uma tabela, a operação de restrição pode ser interpretada como a eliminação das linhas da tabela R que não satisfazem a condição p .

//////////	//////////	//////////
//////////	//////////	//////////

Representação gráfica:



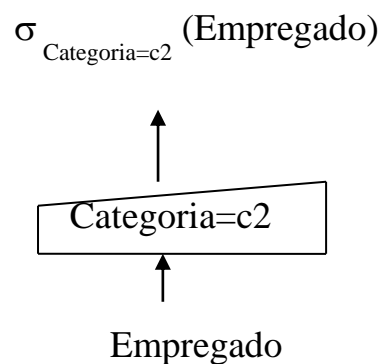
Exemplo: Restrição da tabela *Empregado* tal que *Categoria* = *c2*

$$\sigma_{\text{Categoria} = c2}(\text{Empregado})$$

dá origem à tabela,

Emp#	Nome	Categoria	Dep#
e2	n2	c2	d2
e5	n5	c2	d3
e6	n6	c2	d3

Representação gráfica:



2.2.2.3 Operações com conjuntos

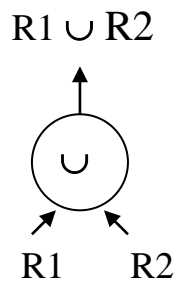
União, Intersecção e Diferença

Dados *R1* e *R2* tais que têm igual número de atributos e os domínios dos atributos correspondentes são os mesmos (esquemas relacionais compatíveis)

A relação resultado tem os mesmos atributos do 1º operando.

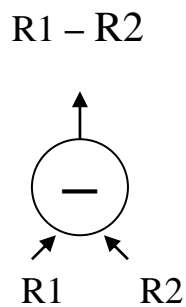
2.2.2.3.1 União

$R1 \cup R2$ é o conjunto dos tuplos de $R1$ e $R2$.



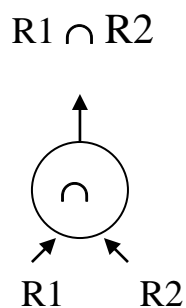
2.2.2.3.2 Diferença

$R1 - R2$ é o conjunto de tuplos de $R1$ que não pertencem a $R2$.



2.2.2.3.3 Intersecção

$R1 \cap R2$ é o conjunto de tuplos comuns a $R1$ e $R2$.

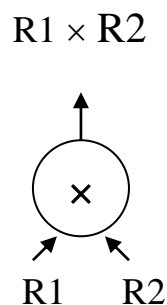


2.2.2.3.4 Produto Cartesiano

Dadas $R1$ e $R2$ com qualquer esquema,

$R1 \times R2$ é concatenação dos atributos de $R1$ e $R2$.

Cada tuplo de $R1$ é concatenado com cada tuplo de $R2$.



2.2.2.4 Operações de junção

2.2.2.4.1 Junção Natural

Sejam $A(Z, X)$ e $B(X, W)$

Junção Natural das relações A e B :

$$A \bowtie B = \{(z, x, w) \mid (z, x) \in A \wedge (x, w) \in B \wedge x = y\}$$

O resultado é uma relação cujo conjunto de atributos são os atributos de A (com os mesmos nomes e domínios) e pelos atributos de B que não aparecem na condição de junção.

Os tuplos da tabela são obtidos pela concatenação dos tuplos de A com os tuplos de B sempre que os valores dos atributos de X sejam iguais.

Exemplo:

Empregados (Emp#, Nome, Categoria, Dep#)

Departamento (Dep#, Nome, Local)

A Expressão:

$$\pi_{\text{Nome, Local}}(\text{Empregado} \bowtie \text{Departamento})$$

denota a composição de duas operações:

- A junção natural das relações *Empregado* e *Departamento* sobre os atributos *Dep#*;
- A projecção do resultado da junção sobre os atributos *Nome* e *Local*.

Nota: assume-se que os atributos Nome (de empregado e de departamento) têm domínios diferentes. Donde não entram na junção natural, pois não são compatíveis em união.

Empregado

Emp#	Nome	Categoria	Dep#
e1	n1	c1	d1
e2	n2	c2	d2
e3	n3	c3	d1
e4	n4	c1	d2
e5	n5	c2	d3
e6	n6	c2	d3
e7	n7	c1	d1

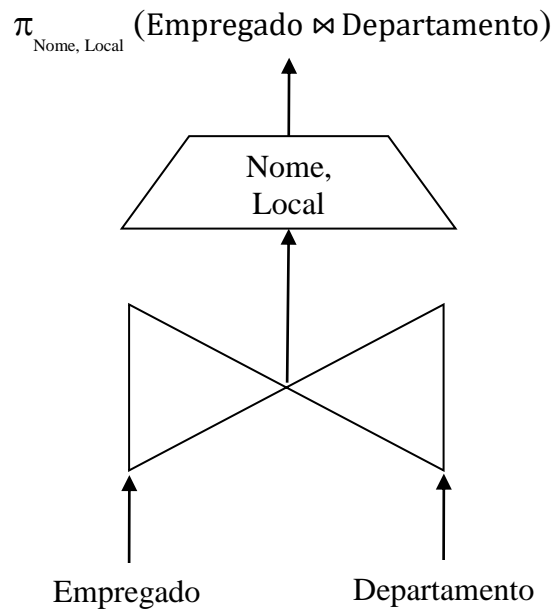
Departamento

Dep#	Nome	Local
d1	N1	l1
d2	N2	l1
d3	N3	l2

Resultado:

Nome	Local
n1	l1
n2	l1
n3	l1
n4	l1
n5	l2
n6	l2
n7	l1

Representação gráfica:



Exercício:

- A que pergunta responde esta operação?

2.2.2.4.2 Equijunção

Sejam as relações $A(Z, X)$ e $B(Y, W)$, com X e Y compatíveis em união.

Equijunção das relações A e B sobre os atributos X e Y :

$$A \bowtie_{X=Y} B = \{(z, x, y, w) \mid (z, x) \in A \wedge (y, w) \in B \wedge x = y\}$$

O resultado é uma relação cujo conjunto de atributos é formado pelos atributos de *A* e de *B*.

Os tuplos da tabela são obtidos pela concatenação dos tuplos de *A* com os tuplos de *B* sempre que os valores dos atributos de *X* são iguais aos valores dos atributos de *Y*.

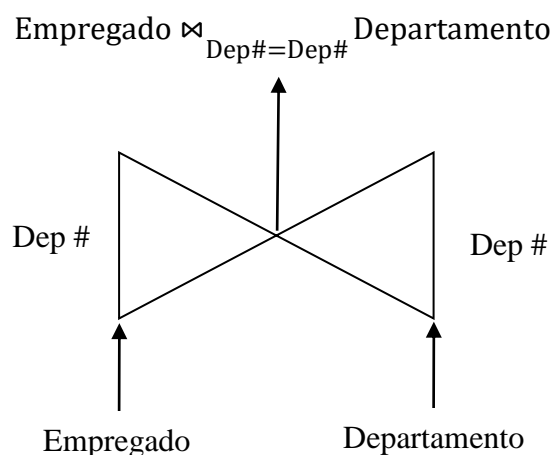
Exemplo:

Obtenha todos os pormenores dos empregados e dos seus departamentos.

Empregado $\bowtie_{\text{Dep\#}=\text{Dep\#}}$ Departamento

Resultado:

Emp#	Empregado.Nome	Categoria	Empregado.Dep#	Departamento.Dep#	Departamento.Nome	Local
e1	n1	c1	d1	d1	N1	l1
e2	n2	c2	d2	d2	N2	l1
e3	n3	c3	d1	d1	N1	l1
e4	n4	c1	d2	d2	N2	l1
e5	n5	c2	d3	d3	N3	l2
e6	n6	c2	d3	d3	N3	l2
e7	n7	c1	d1	d1	N1	l1



2.2.2.4.3 Junção com condições (junção θ)

A junção com condições, ou junção teta, é operação de junção mais geral.

Sejam R e S relações. A θ -junção de R e S é a relação que contém os tuplos do produto cartesiano de R e S que satisfazem a condição F . O predicado F é especificado da forma $R.a_i \theta S.b_i$, onde θ é um dos operadores de comparação ($<$, \leq , $>$, \geq , $=$, \neq) e $R.a_i$ e $S.b_i$ são atributos de R e S , respetivamente.

$$R \bowtie_F S = \sigma_F (R \times S)$$

Nota: quando a condição F contém somente igualdades ($=$) obtemos a equijunção.

Representação gráfica: restrição do produto cartesiano.

2.2.2.4.4 Junção externa (Outer Join)

Por vezes, na junção de tuplos de duas relações verifica-se que um tuplo de uma relação não tem um tuplo correspondente na outra relação. Por outras palavras, não se pode estabelecer a junção. Em certas circunstâncias pode ser útil aparecer os tuplos de uma relação mesmo quando não há valores correspondentes na outra. A junção externa tem esse propósito.

A *junção externa à esquerda* de duas relações R e S , $R \ltimes S$, é uma junção onde os tuplos de R que não tenham correspondência em S também são incluídos na relação resultado. Os valores em falta na segunda relação são colocados a *Null*.

Exemplo:

Junção (natural) externa à esquerda.

Departamento \bowtie Empregado

Empregado

Emp#	Nome	Categoria	Dep#
e1	n1	c1	d1
e2	n2	c2	d2
e3	n3	c3	d1
e4	n4	c1	d2
e5	n5	c2	d3
e6	n6	c2	d3
e7	n7	c1	d1

Departamento

Dep#	Nome	Local
d1	N1	l1
d2	N2	l1
d3	N3	l2
d4	N4	l2

Resultado:

Dep#	Departamento.Nome	Local	Emp#	Empregado.Nome	Categoria
d1	N1	l1	e1	n1	c1
d1	N1	l1	e3	n3	c3
d1	N1	l1	e7	n7	c1
d2	N2	l1	e2	n2	c2
d2	N2	l1	e4	n4	c1
d3	N3	l2	e5	n5	c2
d3	N3	l2	e6	n6	c2
d4	N4	l2	null	null	null

Junção externa à direita: $R \bowtie S$.

Junção externa à esquerda e à direita (full outer join): $R \bowtie S$.

Representação gráfica: similar à junção natural (ou à equijunção).

2.2.2.5 Divisão

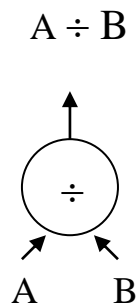
Seja as relações $A(X, Y)$ e $B(Z)$ com Y e Z compatíveis em união.

A divisão de A por B sobre Y e Z é

$$A \div B = \{x \mid \forall z \in B, (x, z) \in A\}$$

Valores de x tais que o par (x, z) ocorre em A para **todos** os valores de z que ocorrem em B .

Representação gráfica:



Exemplo: $\text{Atribuição} \div (\pi_{\text{Proj\#}}(\text{Projeto}))$

Denota a divisão da relação *Atribuição* pela projeção da relação *Projeto* sobre o atributo *Proj#*.

Dadas as relações,

Projeto

Proj#		Designação	Fundos
p1		t1	f1
p2		t2	f2
p3		t3	f3

Atribuição

Emp#	Proj#	Função
e1	p1	r1
e2	p3	r1
e2	p2	r2
e3	p2	r1
e3	p3	r1
e4	p1	r1
e5	p3	r2
e6	p1	r3
e6	p2	r3
e6	p3	r3
e7	p1	r1

O resultado de $\pi_{\text{Proj\#}}$ (Projeto) é

Proj#
p1
p2
p3

O resultado da divisão é:

Emp#	Função
e6	r3

- A que questão responde esta operação?

Exercício:

Dadas as tabelas,

D

S	P
s1	p1
s1	p2
s1	p3
s1	p4
s1	p5
s1	p6
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4
s4	p5

d1

P
p1

d2

P
p2
p4

d3

P
p1
p2
p3
p4
p5
p6

Determine:

a) $D \div d1$; b) $D \div d2$; c) $D \div d3$

2.2.2.6 Renomear (rebatizar)

Por vezes é conveniente usar um operador (ρ , ρ) para explicitamente renomear relações ou atribuir um nome ao resultado de uma operação em álgebra relacional.

$\rho_S(E)$	A operação de renomeação dá um novo nome, S , à
ou	expressão E , e opcionalmente designa os seus
$\rho_{S(A_1, A_2, \dots, A_n)}(E)$	atributos por A_1, A_2, \dots, A_n .

Representação gráfica: similar à representação da projeção.

2.2.2.7 Operações de agregação e de agrupamento⁵

As operações anteriores são as normalmente apresentadas na álgebra relacional clássica. Contudo, por vezes, é necessário efetuar alguma espécie de sumarização ou de agregação de dados, ou alguma forma de agrupamento de dados, o que levou a que novos operadores fossem introduzidos.

Os *operadores de agregação*, tais como a soma ou a média, não são operações da álgebra relacional, mas são usados pelo operador de agrupamento (a seguir). Os operadores de agregação aplicam-se aos atributos (colunas) de uma relação. Por exemplo, a soma de uma coluna produz um número que é a soma dos valores nessa coluna.

⁵ (Garcia-Molina, Ullman, & Widom, 2002), pág. 221.

O agrupamento de tuplos de acordo com os seus valores num ou mais atributos tem o efeito de particionar os tuplos de uma relação em grupos. A seguir, pode ser aplicada a agregação às colunas de cada grupo, dando assim a possibilidade de formular várias *queries* que são impossíveis de formular na álgebra relacional clássica. O *operador de agrupamento* γ permite combinar o efeito de agrupamento e de agregação.

2.2.2.7.1 Operações de agregação

Os operadores de agregação são usados para sumarizar ou agregar os valores de um atributo da relação. Os operadores de agregação standard são:

- COUNT – devolve o número de valores (não necessariamente distintos) de um atributo.
- SUM – devolve a soma dos valores de um atributo.
- AVG – devolve a média dos valores do atributo associado.
- MIN – devolve o menor valor do atributo associado.
- MAX – devolve o maior valor do atributo associado.

Exemplo:

A	B	
1	2	SUM (B) = 2+4+2+2 = 10
3	4	AVG (A) = (1+3+1+1)/4 = 1.5
1	2	MIN (A) = 1
1	2	MAX (B) = 4
1	2	COUNT (A) = 4

2.2.2.7.2 Operação de agrupamento

O **operador γ** permite formar grupos numa relação e/ou agregar colunas. Se houver agrupamento, então a agregação faz-se dentro dos grupos.

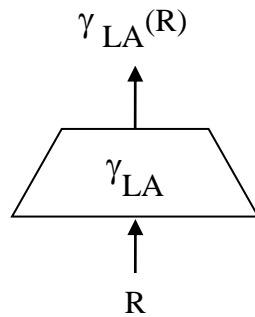
O operador γ tem um índice, uma lista L . Cada elemento de L é um:

- a) Um atributo da relação R ao qual γ é aplicado; este é um dos atributos pelo qual R será agrupada (*atributo de agrupamento*). Ou,
- b) Um operador de agregação aplicado a um atributo da relação (*atributo agregado*). É usada uma seta (\rightarrow) seguida de um nome para fornecer um nome para o valor da agregação.

A relação devolvida pela expressão $\gamma_L(R)$ é construída do modo seguinte:

- 1) Partição dos tuplos de R em grupos. Cada grupo consiste de todos os tuplos contendo o mesmo valor dos atributos agregados da lista L . Se não for especificado qualquer atributo agregado, é considerada toda a relação R como um grupo.
- 2) Para cada grupo é produzido um tuplo consistindo de:
 - i) Os valores dos atributos agregados para esse grupo; e,
 - ii) As agregações, de todos os tuplos do grupo, para os atributos agregados de L .

Representação gráfica (similar à representação da projeção):



Exemplo:

Empregado

Emp#	Nome	Salario	Dep#
e1	n1	1 500	d1
e2	n2	2 700	d2
e3	n3	3 000	d1
e4	n4	2 200	d2
e5	n5	1 750	d3
e6	n6	3 250	d3
e7	n7	1 800	d1

i) *Quantos empregados ganham mais de 2500 u.m?*

$\gamma_{\text{COUNT}_{\text{Emp\#}} \rightarrow \text{Nemps}} (\sigma_{\text{Salario} > 2500} (\text{Empregado}))$

NEmps
3

ii) *Quantos empregados tem cada departamento?*

$\gamma_{\text{Dep\#, COUNT}_{\text{Emp\#}} \rightarrow \text{Nemps}} (\text{Empregado})$

Dep#	NEmps
d1	3
d2	2
d3	2

2.2.2.8 Combinação de operações para formar Queries

A álgebra relacional, tal como as outras álgebras, permite formar expressões complexas, aplicando operadores a relações existentes ou a relações que resultam da aplicação de um ou mais operadores a relações.

As expressões da álgebra relacional podem ser representadas em forma de árvore. Esta representação gráfica permite exprimir questões à Base de Dados que são de fácil leitura (para o operador humano).

Exercícios:

→ Construir a resposta em álgebra relacional.

I1: Quem forneceu o material M1 para a obra O1?

I2: Que materiais (nome) forneceu o fornecedor F2 e para que obras (nome)?

→ A junção não é a uma operação essencial, podendo ser definida em termos de operações mais primitiva. O mesmo é válido para a intersecção e a divisão. As primitivas da linguagem são: União, Diferença, Produto, Seleção e Projeção.

Defina junção, intersecção e divisão em termos dessas 5 primitivas.

Resp.:

$A \div B = \pi_X(A) - \pi_X(\pi_X(A) \times B) - A$, onde $X = T - W$, sendo T o conjunto de atributos de A e W o conjunto de atributos de B .

$$R \cap S = R - (R - S)$$

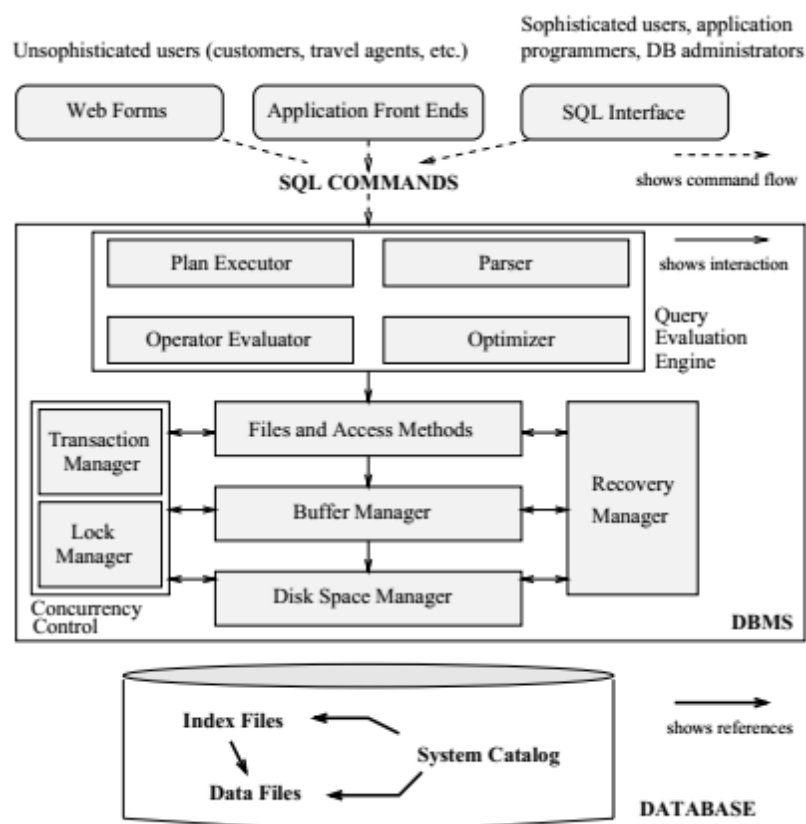
$R \bowtie_F S = \pi_L(\sigma_F(R \times S))$, onde L é a lista formada pelos atributos de R seguidos dos atributos de S que não estão em R .

2.3 SGBD relacional

Note-se que a publicação do modelo relacional precede o aparecimento dos SGBSs, ditos relacionais, em vários anos. Portanto, o modelo é mais abstrato que os sistemas, e a sua existência é completamente independente destes.

2.3.1 Estrutura de um SGBD relacional⁶

O SGBD aceita comandos SQL gerados a partir de uma variedade de interfaces com o utilizador, produz planos de avaliação de consultas, executa esses planos e devolve os resultados.



⁶ (Ramakrishnan & Gehrke, 2003), pág. 19.

Quando um utilizador submete uma consulta, esta é analisada e apresentada a um otimizador de consultas (*query optimizer*), que usa a informação acerca de como os dados estão guardados para produzir um plano de execução eficiente. Um plano de execução (*execution plan*) é um plano para avaliar uma consulta e é geralmente apresentado como uma árvore de operadores relacionais (com anotações que contêm informações detalhadas adicionais sobre quais métodos de acesso usar, etc.). Os operadores relacionais servem de blocos de construção para avaliar as consultas submetidas sobre os dados.

O código que implementa os operadores relacionais assenta sobre a camada de Ficheiros e Métodos de Acesso. Esta camada inclui diverso software para suportar o conceito de ficheiro, que num SGBD, é uma coleção de páginas ou uma coleção de registos. Suporta ficheiros *Heap*, ou ficheiros de páginas não ordenadas, assim como índices. Além disso, para fazer o acompanhamento das páginas dentro do ficheiro, esta camada organiza a informação dentro da página.

A camada Ficheiros e Métodos de Acesso assenta na camada Gestão de Buffers (*buffer manager*), que traz as páginas do disco para a memória principal, conforme necessário, em resposta a solicitações de leitura.

A camada mais baixa do SGBD lida com a gestão de espaço em disco, onde os dados estão armazenados. As camadas de nível superior alocam, desalocam, leem e escrevem páginas através de (funções fornecidas por) esta camada, chamada Gestor de Espaço em Disco (*disk space manager*).

O SGBD suporta a concorrência e a recuperação de falhas através de um rigoroso escalonamento das solicitações dos utilizadores e mantendo um

registo (*log*) de todas as alterações da base de dados. As componentes do SGBD associadas com o controlo de concorrência e recuperação incluem o Gestor de Transações (*transaction manager*), que garante que as transações solicitam e libertam os trincos segundo um protocolo de bloqueio apropriado e agenda a execução das transações; o Gestor de Trincos (*lock manager*), que mantém um registo pedidos de trincos e de concessões de trincos sobre os objetos da base de dados quando estes ficam disponíveis; e o Gestor de Recuperações (*recovery manager*), que é responsável por manter um *log* e por restaurar o sistema para um estado consistente após uma falha. As camadas de gestão de espaço em disco, de gestão de buffers, e de ficheiros e métodos de acesso devem interagir com estas componentes.

2.3.2 Processamento de consultas

“Os objetivos do processamento de consultas são 1) transformar uma consulta escrita numa linguagem de alto nível, tipicamente SQL, numa estratégia de execução correta e eficiente, expressa numa linguagem de baixo nível (implementando a álgebra relacional); e, 2) executar essa estratégia para aceder e obter os dados pretendidos.

Um aspeto importante do processamento de consultas é a otimização. Como podem existir várias transformações equivalentes da mesma consulta de alto nível, o objetivo do otimizador de consultas é escolher aquela que minimiza o uso de recursos. Geralmente, tenta reduzir o tempo de execução da consulta, que é a soma dos tempos de execução de todas as operações individuais que compõem a consulta.”⁷

⁷ (Connolly & Begg, 2015), pág. 729.

As consultas SQL são transformadas numa forma estendida da álgebra relacional, com os planos de avaliação das consultas representados por árvores de operadores relacionais. Portanto, os operadores relacionais servem de blocos construtivos para a avaliação das consultas, sendo a implementação destes blocos cuidadosamente otimizada para se obter um bom desempenho.

A descrição dos dados, ou **metadados**, guardada em tabelas especiais designadas por **catálogo do sistema**, é usada para encontrar a melhor forma de avaliar uma consulta.

2.3.3 Catálogo⁸

Uma tabela pode ser guardada usando uma das várias estruturas de ficheiros alternativas, e para cada tabela podem ser criados um ou mais índices (um índice é uma estrutura de acesso rápido aos dados e está associado a uma ou mais colunas de uma tabela). A coleção de ficheiros correspondentes às tabelas e índices dos utilizadores representam os **dados** da base de dados.

Um SGBD relacional mantém informação acerca de todas as tabelas e índices que contém. Esta informação descritiva é ela própria armazenada numa coleção especial de tabelas, as **tabelas do catálogo**. As tabelas do catálogo são também chamadas **dicionário de dados**, **catálogo do sistema**, ou simplesmente catálogo.

⁸ (Ramakrishnan & Gehrke, 2003), pág. 395.

Informação no Catálogo

O catálogo tem dados globais ao sistema, tais como o tamanho da *buffer pool* e tamanho das páginas, assim como dados sobre as tabelas, índices e vistas, tais como:

- Para cada tabela:
 - O seu *nome da tabela*, *nome e estrutura do ficheiro* onde está armazenada (ex., *heap file*).
 - O *nome e tipo* de cada um dos seus atributos.
 - *Nome* de cada índice na tabela.
 - *Restrições de integridade* (ex., chave primária e chaves estrangeiras).
- Para cada índice:
 - *Nome e estrutura* do índice (ex., árvore *B+*).
 - Atributos da *chave de pesquisa*.
- Para cada vista:
 - O seus *nome e definição*.

Adicionalmente, estatísticas sobre tabelas e índices são guardadas no catálogo e atualizadas periodicamente (não de cada vez que uma tabela é modificada). Frequentemente, são guardadas as informações seguintes:

- Cardinalidade: número de tuplos de cada tabela,
- Tamanho: número de páginas de cada tabela.
- Cardinalidade dos índices: número de valores distintos da chave (de pesquisa) de cada índice.

- Tamanho dos índices: número de páginas de cada índice. (Num índice B+ pode ser o número de folhas)
- Altura do índice: número de níveis não-folha de cada índice.
- Gama do índice: menor valor e maior valor da chave de pesquisa de cada índice.

O catálogo também contém informação acerca dos *utilizadores* e *privilégios*.

Um aspeto interessante dos SGBDs relacionais é que o catálogo é ele próprio armazenado como uma coleção de tabelas.

2.3.4 Fases do processamento de consultas⁹

O processamento de consultas pode ser dividido em quatro fases principais: decomposição (análise e validação), otimização, geração de código e execução.

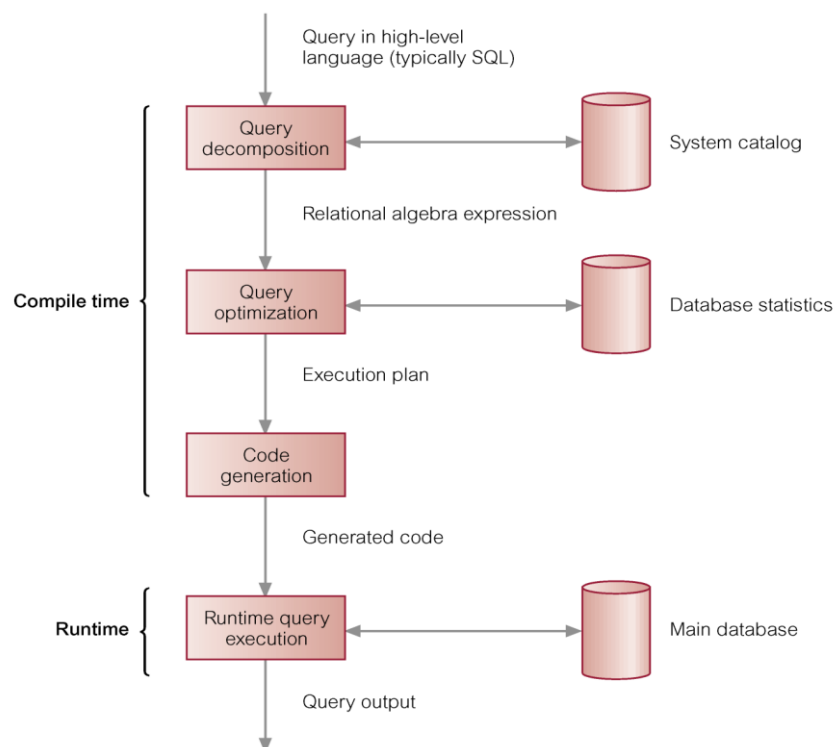
Os objetivos da decomposição de consultas são transformar uma consulta de alto nível numa consulta de álgebra relacional e verificar se a consulta é sintática e semanticamente correta. Os estágios típicos da decomposição são: análise, normalização, análise semântica e reestruturação da consulta.

- *Análise* – nesta etapa, a consulta é lexical e sintaticamente analisada utilizando as técnicas dos compiladores de linguagens de programação. Verifica também se as relações e os especificados na consulta estão definidos no catálogo do sistema. Também verifica se

⁹ (Connolly & Begg, 2015), pág. 731.

as operações aplicadas aos objetos da base de dados são apropriadas para esse tipo de objetos.

- *Normalização* – esta fase converte a consulta numa forma que pode ser facilmente manipulada.
- *Análise semântica* – o objetivo da análise semântica é rejeitar consultas (normalizadas) incorretamente formuladas ou contraditórias. Uma consulta é formulada incorretamente se os componentes não contribuem para a geração do resultado, o que pode acontecer se algumas especificações de junção estiverem em falta. Uma consulta é contraditória se o seu predicado não puder ser satisfeito por algum tuplo.



- *Simplificação* – os objetivos da fase de simplificação são detetar qualificações redundantes, eliminar subexpressões comuns e transformar a consulta em outra semanticamente equivalente, mas

mais fácil e eficiente. Normalmente, são consideradas nesta etapa as restrições de acesso, definições de vistas e restrições de integridade, algumas dos quais também podem introduzir redundância. Se o utilizador não tiver o acesso apropriado a todos os componentes da consulta, a consulta é rejeitada.

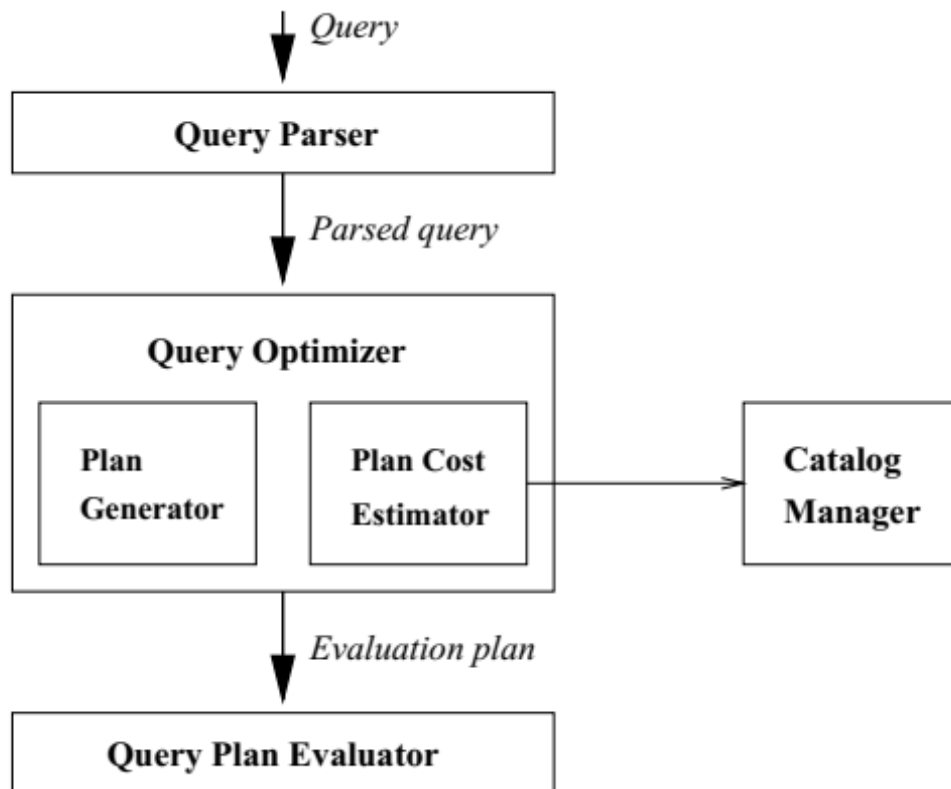
- *Reestruturação da consulta* – na etapa final da decomposição, a consulta é reestruturada para fornecer uma implementação mais eficiente.

2.3.5 Otimização de consultas¹⁰

O objetivo do otimizador de consultas é encontrar um bom plano de avaliação para uma determinada consulta. O espaço de planos considerado por um otimizador de consultas relacionais típico pode ser melhor compreendido ao reconhecer que uma consulta é essencialmente tratada como uma expressão algébrica $\sigma - \pi - \bowtie$, com as operações restantes (se as houver) executadas sobre o resultado da expressão $\sigma - \pi - \bowtie$. Otimizar tal expressão envolve dois passos básicos:

- Enumerar planos alternativos para avaliar a expressão. Tipicamente, o otimizador considera um subconjunto de todos os planos possíveis porque o número de planos possíveis é muito grande.
- Estimar o custo de cada plano enumerado e escolher o plano com o menor custo estimado.

¹⁰ (Ramakrishnan & Gehrke, 2003), pág. 404.



Um plano de avaliação consiste numa árvore de, com anotações adicionais em cada nó, indicando os métodos de acesso a serem usados para cada relação e o método de implementação a ser usado para cada operador relacional.

Considerando a consulta seguinte:

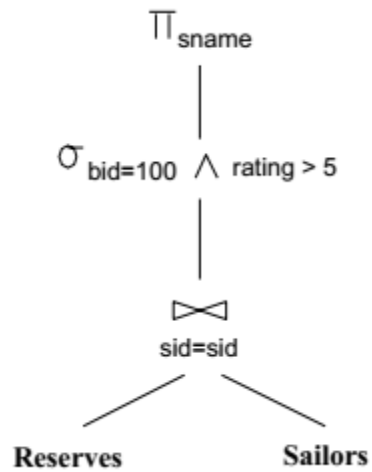
```

SELECT S.sname
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
      AND R.bid = 100 AND S.rating > 5
  
```

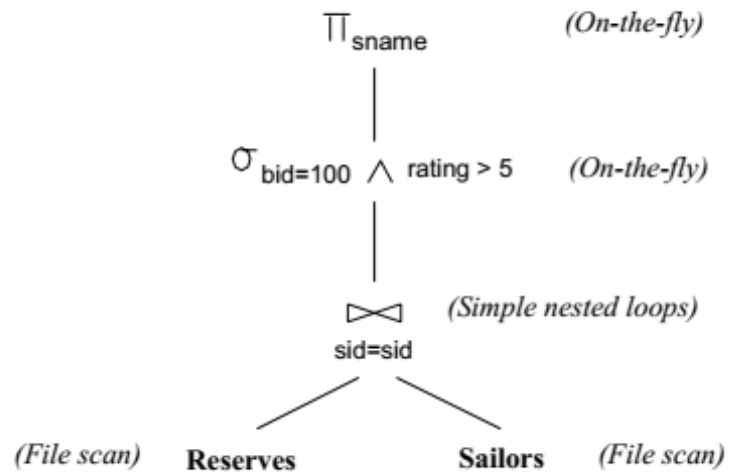
que se pode exprimir em álgebra relacional da forma seguinte:

$$\pi_{\text{sname}}(\sigma_{\text{bid}=100 \wedge \text{rating} > 5}(\text{Reserves} \bowtie_{\text{sid}=\text{sid}} \text{Sailors}))$$

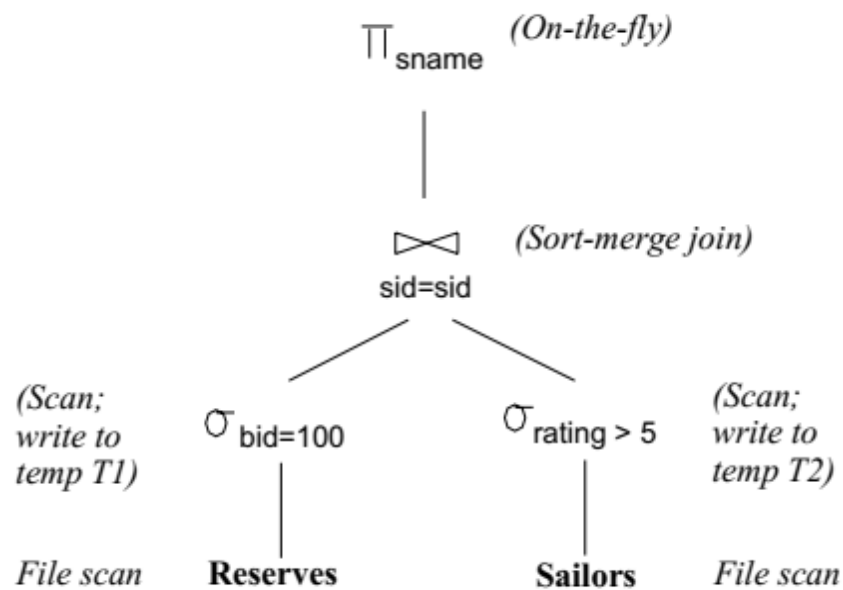
Em termos de representação gráfica (árvore):



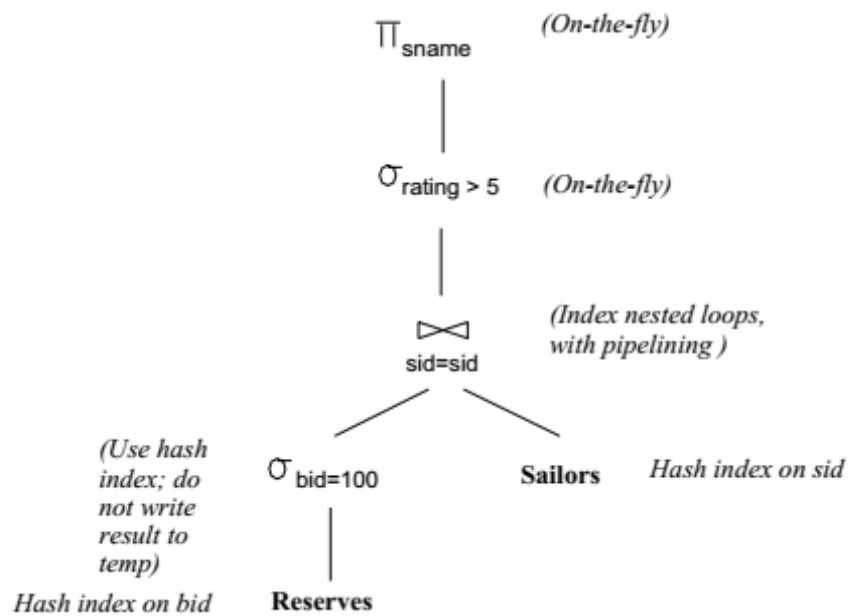
Plano de avaliação:



Outro plano:



Ou ainda outro plano:



2.3.6 As doze regras de Codd¹¹

Codd definiu um conjunto de doze regras a que um SGBD deve obedecer para que possa ser considerado e reconhecido como relacional:

- 1- Numa base de dados relacional, todos os dados, incluindo o próprio dicionário de dados, são representados de uma só forma, em tabelas bidimensionais.
- 2- Cada elemento de dados fica bem determinado pela combinação do nome da tabela onde está armazenado, valor da chave primária e respetiva coluna (atributo).
- 3- Valores nulos (*Nulls*) são suportados para representar informação não disponível ou não aplicável, independentemente do domínio dos respetivos atributos.
- 4- Os metadados são representados e acedidos da mesma forma que os próprios dados.
- 5- Apesar de um sistema relacional poder suportar várias linguagens, deverá existir pelo menos uma linguagem com as seguintes características:
 - Manipulação de dados, com possibilidade de utilização interativa ou em programas de aplicação.
 - Definição de dados.
 - Definição de *views*.
 - Definição de restrições de integridade.
 - Definição de acessos (autorizações).
 - Manipulação de transações (*commit*, *rollback*, etc.).

¹¹ (Pereira, 1998), pág. 176.

- 6- Numa *view*, todos os dados atualizáveis que forem modificados devem ver essas modificações traduzidas nas tabelas base.
- 7- Capacidade de tratar uma tabela (base ou virtual) como se fosse um simples operando (ou seja, utilização de uma linguagem *set-oriented*), tanto em operações de consulta como de atualização.
- 8- Alterações na organização física dos ficheiros da base de dados ou nos métodos de acesso a esses ficheiros (nível interno) não devem afetar o nível conceptual - independência física.
- 9- Alterações no esquema da base de dados (nível conceptual), que não envolvam remoção de elementos, não devem afetar o nível externo - independência lógica.
- 10- As restrições de integridade devem poder ser especificadas numa linguagem relacional, independentemente dos programas de aplicação, e armazenadas no dicionário de dados.
- 11- O facto de uma base de dados estar centralizada numa máquina, ou distribuída por várias máquinas, não deve repercutir-se ao nível da manipulação dos dados.
- 12- Se existir no sistema uma linguagem de mais baixo-nível (tipo *record-oriented*), ela não deverá permitir ultrapassar as restrições de integridade e segurança.

2.4 Linguagens Relacionais

Uma parte importante do modelo de dados é o seu mecanismo de manipulação, ou linguagem de interrogação, que permite recuperar e atualizar os dados existentes. Cood quando definiu o modelo relacional, introduziu também a álgebra relacional e o cálculo relacional como sendo a base para as linguagens relacionais. No entanto, estas linguagens não são muito amigáveis para o utilizador, o que, com base nestas mesmas linguagens, levou ao desenvolvimento de outras linguagens de manipulação de dados. Uma linguagem que emergiu com o desenvolvimento do modelo relacional foi a SQL.

SQL – *Structured Query Language*

Linguagem para o modelo relacional:

- Definida pelo American National Standard Institute (ANSI) em 1986
- Adotada em 1987 como um standard internacional pelo “International Organization for Standardization” (ISO 1987)
- A linguagem SQL possui duas componentes principais:
 - Linguagem de definição de dados (DDL) para definição da estrutura de dados e controlo de acesso.
 - Linguagem de manipulação de dados (DML) para consultar e atualizar os dados.

- Linguagem não procedimental, isto é, especificamos que informação queremos e não como obter essa informação

2.4.1 História do SQL¹²:

SEQUEL (Structured English Query Language), 1974.

SEQUEL/2, 1976. ⇒ SQL (mudança de nome por razões legais).

SQL. Standard ISO em 1987. Não tinha integridade referencial.

SQL.2 ou SQL-92. Publicada uma adenda para contemplar a integridade referencial.

SQL 3 ou SQL-99 – Inclui suporte para gestão de bases de dados orientadas a objetos.

Outras *releases*: 2003 (SQL:2003), 2008 (SQL:2008) e 2011 (SQL:2011).

2.4.2 Query block

(Bloco base de interrogação)

SELECT < lista de atributos>

FROM <lista de relações>

WHERE <expressão lógica>

A estudar detalhadamente nas aulas práticas →

¹² Para mais pormenores consulte-se (Connolly & Begg, 2015), pág. 193.

- Um "*query block*" permite a implementação das operações de seleção, projeção e junção da álgebra relacional.
- Um *query* não especifica a ordem pela qual as operações são executadas.

2) Considere o esquema relacional:

Departamento(DepNum, Nome, Local)

Empregado(EmpNum, Nome, Categoria, Salario, DepNum)

Projecto(ProjNum, Designacao, Fundos)

Atribuicao(EmpNum, ProjNum, Funcao)

2.4.3 Projeção

{ Operação que permite seleccionar tuplos de uma tabela. }

- *O query seguinte constrói uma tabela de números de departamento e categorias de empregados:*

```
Select DepNum, Categoria
From Empregado
```

Nota: Não elimina "linhas" repetidas

```
Select Distinct DepNum, Categoria
From Empregado
```

Nota: Elimina linhas repetidas (pode consumir muito tempo)

{ $\pi_{\text{DepNum, Categoria}}$ (Empregado) }

- *É possível **ordenar** a tabela resultado de um query block:*

```
Select Nome, DepNum
```

From Empregado
Order By Nome

Ordenação por ordem crescente/decrescente:

Select Nome, DepNum
From Empregado
Order By Nome **ASC**, DepNum **DESC**

2.4.4 Restrição

{Operação que permite seleccionar tuplos de uma tabela que satisfazem uma dada condição.}

- *Seleção de todos os empregados que são programadores:*

$\{\sigma_{\text{Categoria}=\text{"Programador"}}(\text{Empregado})\}$

Select *
From empregado
Where Categoria = "Programador"

Nota: Select * → Selecciona todos os atributos.

- ***Seleccção, projecção e ordenação***

i) Nomes dos empregados que são programadores:

Select Nome
From Empregado
Where Categoria = "Programador"
Order By Nome

ii) Nomes dos empregados que são programadores e têm salário superior a 2000€.

$\{\pi_{\text{Nome}}(\sigma_{\text{Categoria}=\text{"Programador"} \wedge \text{Salário} > 2000}(\text{Empregado}))\}$

```
Select Nome
From empregado
Where Categoria = "Programador"
And Salario > 2000
```

iii) *Empregados que trabalham no departamento 7 ou 9.*

$\{ \sigma_{\text{DepNum} = 7 \vee \text{DepNum} = 9}(\text{Empregado}) \}$

```
Select *
From empregado
Where DepNum = 7 or DepNum = 9
```

Equivalente a:

```
Select *
From empregado
Where DepNum In (7, 9)
```

Operadores:

=, <>, >, >=, <, <=
And, Or, Not
In

Exercício:

Qual é o resultado do query seguinte ?

```
Select Nome
From Empregado

Where DepNum In ( Select DepNum
                   From Departamento
                   Where Local = "Lisboa" )
```

R: Nomes dos empregados que pertencem a departamentos localizados em Lisboa.

2.4.5 Junção (Equijunção)

- *Obter uma listagem com o nome dos empregados e a localização dos respectivos departamentos.*

$\{ \pi_{E.Nome, D.Local} (\text{Empregado E} \bowtie_{\text{Dep\#}=\text{DepNum}} \text{Departamento D}) \}$

```
Select E.Nome, D.Local
From Empregado E, Departamento D
Where E.DepNum = D.DepNum
```

Exercício:

- *Qual das consultas seguintes permite fornecer: "Nomes dos programadores e respectivos departamentos se estes estão localizados em Lisboa"?*

(1) Select E.Nome, D.Nome
 From Empregado E, Departamento D
 Where E.Categoria = "Programador"
 And D.Local = "Lisboa"

Ou

(2) Select E.Nome, D.Nome
 From Empregado E, Departamento D
 Where E.Categoria = "Programador"
 And D.Local = "Lisboa"
 And E.DepNum = D.DepNum

R: O segundo query. (Porquê ?)

Supondo,

Empregado

<u>EmpDep</u>	Nome	Categoria	Salário	DepNum
1	E1	Programador	1000	5
2	E2	Programador	1000	6
3	E3	Analista	2000	7

Departamento

<u>DepNum</u>	Nome	Local
5	D1	Lisboa
6	D2	Porto
7	D3	Lisboa

O resultado do query (1) é:

<u>E.Nome</u>	D.nome
E1	Lisboa
E1	Lisboa
E2 !?	Lisboa
E2 !?	Lisboa

O resultado do query (2) é:

<u>E.Nome</u>	D.nome
E1	Lisboa

2.4.6 Produto Cartesiano

Select *
From Empregado, Departamento

2.4.7 União, Intersecção e Diferença

União → **Union**

Intersecção → **Intersect**

Diferença → **Except** (Minus, em alguns casos)

Os operandos têm de ser **compatíveis (em união)**:

- . têm de ter o mesmo grau, i.e., o mesmo número de colunas;
- . colunas correspondentes têm de ter o mesmo domínio.

- *Números dos departamentos que não têm empregados:*

$$\{ \pi_{\text{DepNum}}(\text{Departamento}) - \pi_{\text{DepNum}}(\text{Empregado}) \}$$

Select DepNum
From Departamento

Except

Select DepNum
From Empregado

- *Número de empregado dos programadores que trabalham em algum projeto:*

$$\{ \pi_{\text{EmpNum}}(\sigma_{\text{Categoria} = \text{"Programador"}}(\text{Empregado})) \cap \pi_{\text{EmpNum}}(\text{Atribuicao}) \}$$

Select EmpNum
From Empregado
Where Categoria = "Programador"

Intersect

Select EmpNum
From Atribuicao

2.4.8 Divisão

- *Obter uma tabela com os números de empregado, atribuídos a todos os projetos com fundos superiores a um milhão de euros.*

$$\{ \pi_{\text{EmpNum}, \text{ProjNum}}(\text{Atribuicao}) \div \pi_{\text{ProjNum}}(\sigma_{\text{Fundos} > 1000000}(\text{Projecto})) \}$$

```

Select EmpNum
From Empregado E
Where Not Exists ( Select ProjNum
                   From Projecto
                   Where Fundos > 1000000

                   Except

                   Select ProjNum
                   From Atribuicao
                   Where EmpNum = E.EmpNum
                 )

```

Exercício:

- *Quais as obras (número) que receberam fornecimentos de cimento e de areia?*

2.4.9 Funções Standard

AVG	→ Média
SUM	→ Soma
COUNT	→ Número de elementos
MAX	→ Maior valor
MIN	→ Menor valor

- *Qual o salário médio dos programadores?*

```

Select Avg(Salario)
From Empregado
Where Categoria = "Programador"

```

- *Número de funções diferentes que o empregado 128 executa em projectos:*

```

Select Count(Distinct Funcao)
From Atribuicao
Where EmpNum = 128

```

Nota: Count(*) → Número de tuplos que satisfaz a cláusula *Where*.

2.4.10 Actualizações

A SQL não é só uma linguagem de *query*.

É possível também inserir, eliminar ou modificar tuplos.

2.4.10.1 Inserção

- *Inserir o empregado 843 com o nome José e a categoria Programador:*

Insert Into Empregado (EmpNum, Nome, Categoria)

Values (843,'José','Programador')

. Os atributos não especificados assumem o valor *Null*.

. Se são especificados valores para todos os atributos não é necessário referir os seus nomes (segue a ordem de criação da tabela).

Supondo que além das relações anteriores existe também a relação:

Candidatos (*EmpNum*, Nome, Categoria, Salário, DepNum)

O comando

Insert into Empregado (EmpNum, Nome, Categoria, Salario, DepNum)

Select EmpNum, Nome, Categoria, Salário*1.1, DepNum

From Candidatos

Where Categoria In (“Programador”, “Analista”)

insere na relação Empregado os tuplos seleccionados da relação Candidatos.

2.4.10.2 Eliminação

- *Eliminação do tuplo do empregado 843:*

Delete From Empregado

Where EmpNum = 843

- *Eliminar todos os empregados cujo departamento está localizado em "Lisboa":*

Delete From Empregado

Where DepNum IN (Select DepNum
 From Departamento
 Where Local = "Lisboa"
)

2.4.10.3 Actualização

- *Aumentar o salário (em 40%) do empregado 843:*

Update Empregado

Set Salario = Salario * 1.4

Where EmpNum = 843

- *Aumentar o salário (em 20%) aos empregados do projeto 10:*

Update Empregado

Set Salario = Salario * 1.2

Where EmpNum IN (Select EmpNum
 From Atribuicao
 Where ProjNum = 10
)

2.5 Restrições de integridade

Uma base de dados está num estado de integridade se contém apenas dados válidos.

Os dados armazenados devem estar de acordo com a realidade.

Empregado (Emp#, Nome, Categoria, Salário, Dep#)

<u>Emp#</u>	Nome	Categoria	Salário	Dep#	Data_Nasc
1	António Sousa	Programador	-1000	5	20-03-1980
2	Ana Amaral	Programador	1000	6	22-03-1970
3		Analista	2000	7	12-04-1964
4	Carlos Silva	Operador	1	5	20-08-2060

Diagram illustrating data integrity constraints on the 'Empregado' table:

- Salário: Não pode ser negativo (Cannot be negative)
- Nome: O campo não pode ser nulo (The field cannot be null)
- Salário: Demasiado pequeno (Too small)
- Data_Nasc: Data inválida (Invalid date)

Restrições de integridade são regras, que definem a validade dos dados

Por exemplo, para a relação anterior:

- O campo Nome não pode ser nulo
- O Salário tem de ser superior ao valor do salário mínimo nacional
- A Data de nascimento tem de ser maior que 01-01-1920 e menor que 01-01-2019!!

. Estas regras vão fazer parte da definição da tabela.

. Quando um dado é inserido, alterado ou apagado, o SGBD vai verificar se as regras definidas são respeitadas.

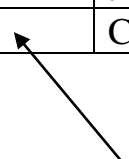
As regras do exemplo anterior denominam-se restrições de integridade de domínio.

2.5.1 Integridade de domínio

São regras que se aplicam aos atributos de uma dada tabela, definindo o domínio de cada atributo.

2.5.2 Integridade de entidade

<u>Emp#</u>	Nome	Categoria	Salário	Dep#	Data_Nasc
1	António Sousa	Programador	1000	5	20-03-1980
2	Ana Amaral	Programador	1000	6	22-03-1970
3	José Costa	Analista	2000	7	12-04-1964
2	Carlos Silva	Operador	500	5	20-08-1980



Um campo que é chave primária não pode ter valores duplicados (nem ter valor nulo)

- Ao declararmos um atributo como chave primária da relação o SGBD não deixa que a relação tenha dois tuplos com o mesmo valor nesse atributo

2.5.3 Integridade referencial

Restrição de integridade que relaciona duas relações

Empregado

<u>Emp#</u>	Nome	Categoria	Salário	Dep#	Data_Nasc
1	António Sousa	Programador	1000	5	20-03-1980
2	Ana Amaral	Programador	1000	6	22-03-1970
3	José Costa	Analista	2000	7	12-04-1964
4	Carlos Silva	Operador	500	5	20-08-1980

Departamento

<u>Dep#</u>	Nome	Local
5	D1	Lisboa
6	D2	Porto
7	D3	Lisboa

O atributo *Dep#* na tabela *Empregado* é chave estrangeira (ou externa) sendo chave primária na tabela *Departamento*

Se, se indica que *Dep#* é chave estrangeira da relação *Empregado* então cada valor do atributo *Dep#* na tabela *Empregado* tem obrigatoriamente de existir na tabela *Departamento*.

O que acontece quando se tenta apagar na tabela *Departamento* o departamento cujo *Dep#* = 5?

- Ou o SGBD não deixa apagar;
- Ou apaga o registo e depois apaga na tabela *Empregado* todos os empregados cujo número de departamento é 5 (apagamento em cascata)

2.5.4 Regras de negócio

Restrições de integridade mais complexas que não podem ser definidas na estrutura da base de dados. São verificadas pelos programas de aplicação.

Exemplos

- . O salário de um empregado não pode diminuir, só aumentar
- . Um empregado não pode ganhar mais do que o seu chefe
- ...

3 Teoria da Normalização

Ao modelar a informação procura-se:

- . Um modelo que represente fielmente a realidade.
- . Um modelo capaz de responder às funcionalidades que se pretendem.

Queremos obter um modelo com propriedades que garantam:

- . Redundância mínima
- . Facilidade de Manutenção
- . Estabilidade face a futuras alterações

3.1 Dados redundantes

EmpregadoDepartamento

<u>NumEmp</u>	Nome	Categoria	Salário	Dep	TelDep	LocalDep
10	José da Silva	Programador	2500	1	213334555	Lisboa
20	Maria Costa	Analista	5000	2	224446888	Porto
30	João Fonseca	Operador	600	1	213334555	Lisboa
40	Ana Faria	Analista	5200	4	275222333	Covilhã

Nesta tabela existem dados redundantes. Os dados de um dado departamento são repetidos para cada empregado desse departamento.

Se apagarmos este número de departamento deixamos de saber qual o departamento do empregado 30.
Dado **duplicado** mas não redundante

Dados redundantes.
Se apagarmos esta informação continuamos a saber os dados do departamento 1!!

Relações que têm dados redundantes podem vir a ter anomalias de inserção, eliminação ou modificação.

Anomalias de inserção:

. Para inserir um novo empregado temos que inserir toda a informação do departamento a que pertence tendo o cuidado de não criar inconsistências com a informação já existente.

. Não é possível criar um departamento que ainda não tenha empregados. Note-se que o atributo *NumEmp* é chave da relação logo o seu valor não pode ser nulo.

Anomalias de eliminação:

. Se eliminarmos um empregado que seja o único empregado de um dado departamento perdemos a informação desse departamento.

Anomalias de modificação:

. Se quisermos alterar um atributo de um dado departamento (por ex., o telefone do dep. 1) temos de atualizar o valor do atributo em todos os empregados que pertencem a esse departamento.

Podemos evitar as anomalias anteriores se decompusermos a relação *EmpregadoDepartamento* nas relações:

Empregado (*NumEmp*, *Nome*, *Categoria*, *Salário*, *Dep*) e
Departamento (*Dep*, *TelDep*, *LocalDep*).

A teoria da normalização, desenvolvida por Edgar Codd no âmbito do modelo relacional, define um processo de estruturar as tabelas de uma base de dados de forma a minimizar a redundância de dados.

A teoria da normalização utiliza o conceito da Dependência Funcional que vamos começar por estudar:

3.2 Dependências Funcionais

Seja a relação:

Morada (Nome, Endereço, Cidade, CodPostal, Telefone)

Assumindo que todos os nomes são diferentes, podemos descrever as seguintes dependências entre os atributos da relação *morada*:

- i) Dado um Nome específico, este determina um único tuplo da relação *Morada*
(isto é, determina valores únicos para os atributos Endereço, Cidade, CodPostal, Telefone)
- ii) Dados valores dos atributos Endereço e Cidade, todos os tuplos da relação *Morada* com esses valores (se existirem) têm o mesmo valor do atributo CodPostal.
- iii) A um determinado valor de *CodPostal* corresponde um único valor do atributo Cidade.

As associações lógicas descritas entre os atributos da relação *Morada* são denominadas **dependências lógicas**.

Tipos de dependências lógicas:

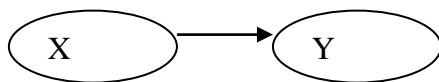
- . Dependências Funcionais
- . Dependências Multivalor
- . Dependências de Junção

3.2.1 Definição de Dependência Funcional (DF)

Seja $R(A_1, A_2, \dots, A_n)$ um esquema de relação e X e Y subconjuntos de $\{A_1, A_2, \dots, A_n\}$ não vazios.

Existe uma Dependência Funcional entre X e Y , $X \rightarrow Y$ (X determina Y) sse **em qualquer instante t** , quaisquer tuplos de R com o mesmo valor de X têm necessariamente o mesmo valor de Y .

3.2.2 Diagrama de Dependência Funcional



- . X determina Y
- . Y depende de X

(1) Exemplo

$\text{Nome} \rightarrow \text{Endereço}$

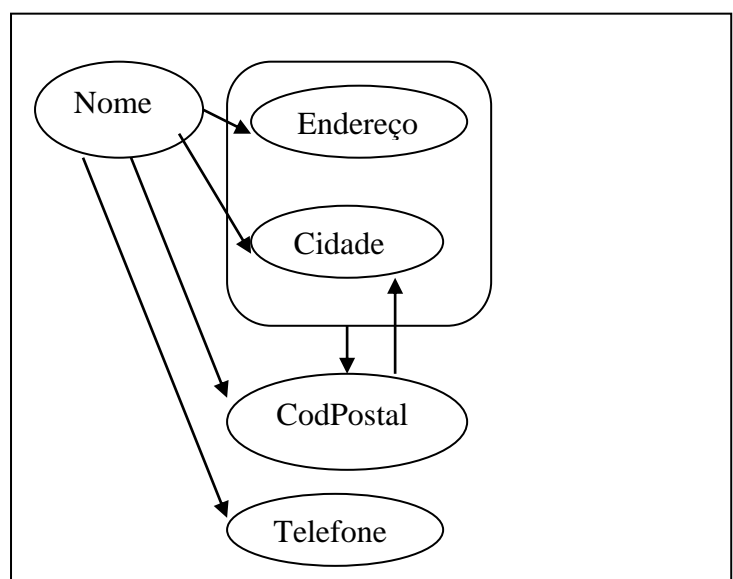
$\text{Nome} \rightarrow \text{Cidade}$

$\text{Nome} \rightarrow \text{CodPostal}$

$\text{Nome} \rightarrow \text{Telefone}$

$\text{Endereço, Cidade} \rightarrow \text{CodPostal}$

$\text{CodPostal} \rightarrow \text{Cidade}$



3.2.3 Chave (candidata)

Seja a relação $R (A_1, A_2, \dots, A_n)$ e X um conjunto de atributos de R

$$(X \subseteq \{A_1, A_2, \dots, A_n\})$$

X é chave de R sse

$$i) \forall_i X \rightarrow A_i, i = 1, 2, \dots, n$$

X determina funcionalmente todos os atributos de R

$$ii) \nexists Y \subsetneq X: \forall_i Y \rightarrow A_i, i = 1, 2, \dots, n$$

Não existe um subconjunto próprio de X que determine todos os atributos de R

→ *Nome é a única chave da relação morada*

3.2.4 Superchave

Qualquer conjunto de atributos X que satisfaz a condição $i)$ é denominado superchave da relação.

→ *Qualquer conjunto de atributos que contenha o atributo Nome é superchave da relação morada*

3.2.5 Toda a relação tem uma chave

Demonstração:

Dada uma relação $R (A_1, A_2, \dots, A_n)$ verifica-se que

$$A_1, A_2, \dots, A_n \rightarrow A_i, i = 1, 2, \dots, n$$

Se não existe um $X \subsetneq \{A_1, A_2, \dots, A_n\}$ tal que $X \rightarrow A_i$ então A_1, A_2, \dots, A_n é a chave de R .

Caso contrário X contém uma chave.

3.2.6 Chave Primária

- É a chave candidata escolhida.
- Nenhuma das suas ocorrências pode ter valor nulo.

3.2.7 Propriedades básicas das DFs

3.2.7.1 Unicidade

- Se $f: X \rightarrow Y$ e $g: X \rightarrow Y$ então $f = g$.

3.2.7.2 Reflexibilidade

- Se $X \supseteq Y$ então $X \rightarrow Y$.

3.2.7.3 Transitividade

- Se $X \rightarrow Y$ e $Y \rightarrow Z$ então $X \rightarrow Z$.

3.2.7.4 Aumento

- Se $X \rightarrow Y$ então $XZ \rightarrow YZ$, para qualquer Z .

3.2.7.5 Axiomas de Armstrong

As regras de inferência, reflexibilidade, aumento e transitividade, são conhecidas por axiomas de Armstrong.

Exemplo:

Voltando à relação Morada,

Morada (Nome, Endereço, Cidade, CodPostal, Telefone).

DF's:

$Nome \rightarrow Endere\c{c}o$

$Nome \rightarrow Cidade$

$Nome \rightarrow CodPostal$

$Nome \rightarrow Telefone$

$Endere\c{c}o, Cidade \rightarrow CodPostal$

$CodPostal \rightarrow Cidade$

A – Por reflexibilidade,

$Endere\c{c}o, Cidade \rightarrow Endere\c{c}o$

$Endere\c{c}o, Cidade \rightarrow Cidade$ [1]

B – De “ $Endere\c{c}o, Cidade \rightarrow CodPostal$ ” e “ $CodPostal \rightarrow Cidade$ ”,

por transitividade obtemos

$Endere\c{c}o, Cidade \rightarrow Cidade$ [2]

C – Pela unicidade, [1] e [2], são a mesma dependência funcional.

3.2.8 Propriedades derivadas das DFs

Aplicando o conjunto de axiomas de Armstrong podem derivar-se algumas regras adicionais das DFs:

3.2.8.1 Distributividade (decomposição)

Se $X \rightarrow YZ$ então $X \rightarrow Y$ e $X \rightarrow Z$.

Dem.

- a) $X \rightarrow YZ \Rightarrow YZ \rightarrow Y \wedge YZ \rightarrow Z$ {hipótese; reflexibilidade, reflexibilidade}
- b) $X \rightarrow YZ \wedge YZ \rightarrow Y \Rightarrow X \rightarrow Y$ {hipótese; a); transitividade}
- c) $X \rightarrow YZ \wedge YZ \rightarrow Z \Rightarrow X \rightarrow Z$ {hipótese; a); transitividade}

3.2.8.2 União

Se $X \rightarrow Y$ e $X \rightarrow Z$ então $X \rightarrow YZ$.

Dem.

- a) $X \rightarrow Z \Rightarrow X \rightarrow ZX$ {hipótese, aumento}
- b) $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$ {hipótese, aumento}
- c) $X \rightarrow XZ \wedge XZ \rightarrow YZ \Rightarrow X \rightarrow YZ$ {a), b), transitividade}

3.2.8.3 Pseudotransitividade

Se $X \rightarrow Y$ e $YW \rightarrow Z$ então $XW \rightarrow Z$.

Dem.

- a) $X \rightarrow Y \wedge W \rightarrow W \Rightarrow XW \rightarrow YW$ {hipótese, reflexibilidade, aumento}
- b) $XW \rightarrow YW \wedge YW \rightarrow Z \Rightarrow XW \rightarrow Z$ {a), hipótese, transitividade}

Exemplo:

Aplicando a pseudotransitividade às dependências funcionais

$Nome \rightarrow Endere\c{c}o$ e $Endere\c{c}o, Cidade \rightarrow CodPostal$

obtemos $Nome, Cidade \rightarrow CodPostal$

3.2.9 DF trivial

Dependência Funcional Trivial: todo o conjunto de atributos determina todos os seus subconjuntos.

Notas:

- a) Numa DF trivial, o “lado direito” contém somente atributos que também aparecem no “lado esquerdo”;
- b) Se o atributo dependente não pertence ao determinante, a dependência é não-trivial.

3.2.10 Fecho de um conjunto de DFs

Diz-se que a DF f é *implícada* por um dado conjunto F de DFs se f é válida em cada instância da relação que satisfaz as dependências de F , i.e., f é válida sempre que todas as DFs de F se verificam.

Ao conjunto das DFs implicadas por um dado conjunto F de DFs é chamado **fecho de F** , denotado com F^+ .

Os axiomas de Armstrong podem ser aplicados repetidamente para inferir todas as DFs implicadas por um conjunto F de DFs.

Os axiomas de Armstrong são **sólidos**, pois só geram DFs válidas, i.e., pertencentes a F^+ , quando aplicados a um conjunto F de FDs. São também **completos**, pois a aplicação repetida das regras gerará todas as DFs de F^+ .

Algoritmo para calcular o fecho F^+ :

```

 $F^+ = F;$ 
Do forever {
  For each  $f \in F^+$  {
    Aplicar as regras de reflexibilidade e aumento a  $f$ .
    Adicionar as DFs resultantes a  $F^+$ 
  }
  For each  $f_1, f_2 \in F^+$  {
    If  $f_1$  e  $f_2$  puderem ser combinadas por transitividade Then
      Adicionar a DF resultante a  $F^+$ 
    }
  If  $F^+$  não se alterou nesta iteração Then
    break
}

```

3.2.11 Fecho de um conjunto de atributos

O fecho de um conjunto de atributos X^+ , tendo em conta o conjunto F de DFs, é o conjunto de atributos A tais que $X \rightarrow A$ pode ser inferido aplicando os axiomas de Armstrong.

Algoritmo para calcular o fecho de um conjunto de atributos X por F :

```

 $X^+ = X;$ 
Do forever {
  For each DF  $U \rightarrow V$  in  $F$  {
    If  $U \subseteq X^+$  Then
       $X^+ = X^+ \cup V$ 
    }
  If  $X^+$  não se alterou nesta iteração Then
    break
}

```

Exemplo:

Sejam $R(A, B, C, D, E, F)$ e $F = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$.

Determinar $\{A, B\}^+$:

1. Seja $X^+ = \{A, B\}$.
2. Como ambos os atributos no lado esquerdo da DF $AB \rightarrow C$ pertencem a X^+ , o atributo C , que está do lado direito, pode ser adicionado a X^+ , ficando $X^+ = \{A, B, C\}$.
3. A seguir, como o lado esquerdo de $BD \rightarrow AD$ pertence a X^+ , então pode adicionar-se A e D a X^+ , ficando $X^+ = \{A, B, C, D\}$.

4. Pode-se aplicar o mesmo raciocínio para $D \rightarrow E$, levando a $X^+ = \{A, B, C, D, E\}$.
5. Não são possíveis mais alterações a X^+ . Em particular, a DF $CF \rightarrow B$ não pode ser usada, pois o seu lado esquerdo nunca fica contido em X^+ .
6. Portanto, $X^+ = \{A, B, C, D, E\}$.

3.2.12 Cobertura e equivalência¹³

Sejam S_1 e S_2 dois conjuntos de DFs. Se qualquer DF implicada por S_1 também é implicada por S_2 , ou seja, se S_1^+ é um subconjunto de S_2^+ , diz-se que S_2 é uma **cobertura** de S_1 .

Se S_2 é uma cobertura de S_1 e S_1 é uma cobertura de S_2 , i.e., se $S_1^+ = S_2^+$, diz-se que S_1 e S_2 são **equivalentes**.

Um conjunto S de DFs é **irredutível** sse satisfaz as seguintes propriedades:

- O lado direito (dependente) de qualquer DF em S contém apenas um atributo (i.e., é um conjunto singular);
- O lado esquerdo (determinante) de qualquer DF em S é irredutível por seu lado – o que significa que nenhum atributo pode ser retirado do determinante sem alterar o fecho S^+ (i.e., sem converter S num conjunto que não é equivalente a S). Diz-se que tal DF é irredutível à esquerda.
- Nenhuma DF em S pode ser removida de S sem alterar o fecho S^+ (i.e., sem converter S num conjunto não equivalente a S).

¹³ (Date, 2004), pág. 341.

Para todo o conjunto de DFs existe pelo menos um conjunto equivalente que é irreduzível. Pode existir mais do que uma cobertura irreduzível (ou mínima) para um dado conjunto de DFs.

Exemplo:

Considere-se $R(A, B, C, D)$ e o conjunto F de DF, $F = \{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C, AC \rightarrow D\}$.

Determine um conjunto de DF irreduzível e equivalente a F .

1. O primeiro passo é reescrever as DFs com um elemento singular do lado direito:

$A \rightarrow B$
 $A \rightarrow C$
 $B \rightarrow C$
 $A \rightarrow B$
 $AB \rightarrow C$
 $AC \rightarrow D$

Observa-se que a DF $A \rightarrow B$ ocorre duas vezes, portanto uma ocorrência é eliminada.

2. A seguir, o atributo C pode ser eliminado do lado esquerdo da DF $AC \rightarrow D$, pois tem-se $A \rightarrow C$, donde $A \rightarrow AC$ por aumento, e como $AC \rightarrow D$, $A \rightarrow D$ por transitividade; portanto, C do lado esquerdo de $AC \rightarrow D$ é redundante.
3. A DF $AB \rightarrow C$ pode ser eliminada, porque, novamente, $A \rightarrow C$, donde $AB \rightarrow CB$ por aumento e $AB \rightarrow C$ por decomposição.
4. Finalmente, a DF $A \rightarrow C$ é implicada pela DF $A \rightarrow B$ e $B \rightarrow C$, donde também pode ser eliminada. Fica então:
 $A \rightarrow B$
 $B \rightarrow C$
 $A \rightarrow D$.

Este conjunto é irreduzível.

3.2.13 Chaves e Fecho

Note-se que o conjunto $\{A_1, A_2, \dots, A_n\}^+$ é o conjunto de todos os atributos de uma relação sse A_1, A_2, \dots, A_n for uma superchave da relação.

O algoritmo para determinar o fecho de um conjunto de atributos pode ser usado para determinar as chaves de uma relação, começando com X contendo um atributo singular e parando assim que o fecho contiver todos os atributos da relação. Variando o atributo inicial e a ordem em que o algoritmo considera as DFs, podem obter-se todas as chaves candidatas.

3.2.14 Perda de informação

Seja a relação

$R(\text{Nome}, \text{Telefone}, \text{Cidade})$

em que os atributos *Telefone* e *Cidade* não dependem funcionalmente do atributo *Nome*.

$\text{Nome} \nrightarrow \text{Telefone}$

$\text{Nome} \nrightarrow \text{Cidade}$

(isto é, uma pessoa pode ter mais que um telefone numa ou mais cidades)

Num dado instante podemos ter;

R

Nome	Telefone	Cidade
José da Silva	123456789	Leiria
José da Silva	222222222	Faro
António Costa	333333333	Leiria

As projeções

$$R1 = \pi_{\text{Nome, Telefone}}(R) \quad \text{e} \quad R2 = \pi_{\text{Nome, Cidade}}(R),$$

no mesmo instante de tempo, teriam como resultado:

R1

Nome	Telefone
José da Silva	123456789
José da Silva	222222222
António Costa	333333333

R2

Nome	Cidade
José da Silva	Leiria
José da Silva	Faro
António Costa	Leiria

A junção das duas projeções sobre o atributo *Nome*, $R1 \bowtie R2$, é representada pela tabela

Nome	Telefone	Cidade
José da Silva	123456789	Leiria
José da Silva	123456789	Faro
José da Silva	222222222	Leiria
José da Silva	222222222	Faro
António Costa	333333333	Leiria



A junção da decomposição não é igual à relação inicial!!!

A relação não pode ser decomposta desta forma!

3.2.15 Decomposição sem perda (*Lossless Join*)

Seja $R(X, Y, Z)$ com X , Y e Z conjuntos de atributos.

Se $X \rightarrow Y$ ou $X \rightarrow Z$ então $R(X, Y, Z) = \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$

Dem:

$$1) R(X, Y, Z) \subseteq \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R):$$

Seja $x y z$ um tuplo de $R(X, Y, Z)$.

Se $x y z \in R(X, Y, Z)$ então $x y \in \pi_{X, Y}(R)$ e $x z \in \pi_{X, Z}(R)$

Donde, a junção natural dos tuplos $x y$ e $x z$, sobre o atributo x , origina o tuplo $x y z$. Portanto, $x y z \in \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$.

$$2) \text{ Se } X \rightarrow Y \text{ ou } X \rightarrow Z \text{ então } R(X, Y, Z) \supseteq \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R):$$

Admitindo que $X \rightarrow Y$.

(*neste caso, se $x y z \in R(X, Y, Z)$ e $x y' z' \in R(X, Y, Z)$ então $y = y'$*)

Se $x y z \in \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$ então $\exists y', z'$:

$$x y z' \in R(X, Y, Z) \wedge x y' z \in R(X, Y, Z)$$

Mas como, por hipótese, $X \rightarrow Y$ então $y = y'$. Logo, $xyz \in R(X, Y, Z)$.

Ou seja, $\pi_{X, Y}(R) \bowtie \pi_{X, Z}(R) \subseteq R(X, Y, Z)$.

Portanto, de 1) e 2):

$$R(X, Y, Z) \subseteq \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$$

e

$$R(X, Y, Z) \supseteq \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$$

Donde, $R(X, Y, Z) = \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$. **q.e.d.**

3.3 Normalização

A normalização de uma relação é obtida pela sua decomposição em duas ou mais relações de acordo com um procedimento bem definido.

Três níveis de normalização foram definidos por Codd:

1ª Forma Normal

2ª Forma Normal

3ª Forma Normal

Posteriormente, R. Boyce e Codd (Codd 1974) definiram a

Forma Normal de Boyce-Codd (FNBC)

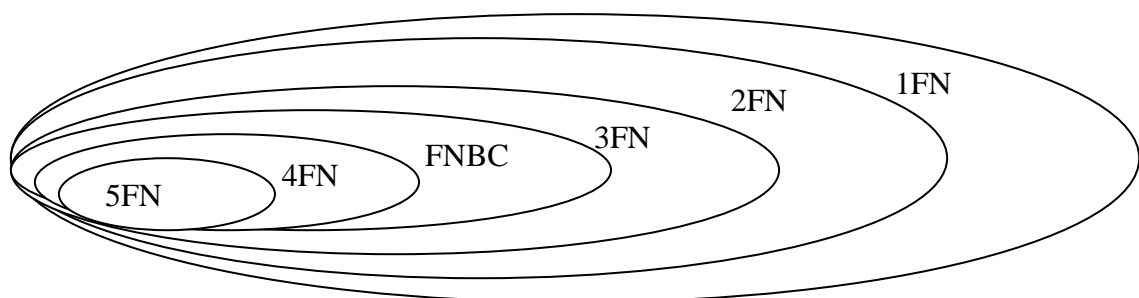
Mais tarde Fagin (1979) propôs:

4ª Forma Normal

5ª Forma Normal

. Uma relação numa forma normal mais avançada tem menos dados redundantes.

. Se uma relação está numa forma normal mais avançada também está nas formas normais anteriores.



3.3.1 Primeira Forma Normal (1FN)

Uma relação $R(A_1, A_2, \dots, A_n)$ é designada por *relação universal* se contiver todos os atributos relevantes da organização. A relação universal garante que todos os atributos têm nomes distintos.

Uma relação está na **1ª Forma Normal** se

- . Cada atributo contém apenas valores atômicos.
- . Não há conjuntos de atributos repetidos descrevendo a mesma característica.

Exemplo de relações que não estão em 1FN:

1)

PessoaCursos1

Nome	Endereço	NIF	Cursos
Artur	Covilhã	123456789	Programador
Ana	Fundão	222222222	Operador, Programador
Carlos	Covilhã	222333444	Analista, Programador, Operador
Paulo	Guarda	555666777	Operador, Analista

- O atributo Cursos **contém valores não atômicos!!!**

2)

PessoaCursos2

Nome	Endereço	NIF	Curso1	Curso2	Curso3
Artur	Covilhã	123456789	Programador		
Ana	Fundão	222222222	Operador	Programador	
Carlos	Covilhã	222333444	Analista	Programador	Operador
Paulo	Guarda	555666777	Operador	Analista	

- São repetidos atributos do mesmo tipo: *Curso1*, *Curso2* e *Curso3*.
(Diz-se que a relação tem um **grupo repetitivo**)
- Os tuplos correspondentes a alunos com apenas 1 ou dois cursos vão ter valores nulos para alguns atributos.
- Como representar uma pessoa com mais do que três cursos?

Suponhamos a relação,

$R(N_nota_enc, Cod_cliente, Nome_cliente, Morada_cliente,$
 $(Cod_produto, Desc_produto, Preço_produto, Quantidade)^*)$

* - Os dados de cada produto encomendado (isto é, de cada linha da nota de encomenda) constituem um grupo de atributos que se repete.

Como decompor a relação?

A uma nota de encomenda corresponde um único cliente (nº e nome) e uma única morada de cliente. Isto é, existe a Dependência Funcional,

$N_nota_enc \rightarrow Cod_cliente, Nome_cliente, Morada_cliente$

Podemos decompor a relação R nas relações:

$Nota_de_enc$ (N_nota_enc , $Cod_cliente$, $Nome_cliente$, $Morada_cliente$)

e

$Linha_nota_enc$ (N_nota_enc , $Cod_produto$, $Desc_produto$,
 $Preço_produto$, $Quantidade$)

A chave da relação $Nota_de_enc$ é N_nota_enc .

A chave da relação $Linha_nota_enc$ é N_Nota_enc , $Cod_produto$.

Ambas as relações estão na 1ª Forma Normal (não têm grupos repetitivos).

3.3.2 Segunda Forma Normal (2FN)

Seja a relação R ($Fornecedor$, $Peça$, $Cidade$, $Quantidade$), onde:

$Fornecedor \leftrightarrow Peça$

$Peça \leftrightarrow Fornecedor$

$Fornecedor \rightarrow Cidade$

Num dado instante t ,

R

<u>Fornecedor</u>	<u>Peça</u>	Cidade	Quantidade
Empresa A	1	Covilhã	100
Empresa A	2	Covilhã	200
Empresa A	3	Covilhã	300
Empresa B	1	Fundão	400
Empresa B	3	Fundão	500

Algumas anomalias:

Inserção: *Não é possível inserir um fornecedor sem que ele forneça alguma peça (Peça faz parte da chave).*

Eliminação: *Se, por exemplo, a empresa B deixar de fornecer as peças 1 e 3, perdemos a informação sobre a cidade desse fornecedor.*

Modificação: *Supondo que um fornecedor muda de cidade. Atualizar R significa atualizar todos os tuplos desse fornecedor.*

Se substituirmos R por

$$R1 = \pi_{\text{Fornecedor, Cidade}}(R) \qquad R1(\underline{\text{Fornecedor}}, \text{Cidade})$$

$$R2 = \pi_{\text{Fornecedor, Peça, Quantidade}}(R) \qquad R2(\underline{\text{Fornecedor, Peça}}, \text{Quantidade})$$

desaparecem as anomalias.

A DF $\text{Fornecedor} \rightarrow \text{Cidade}$ garante que $R = R1 \bowtie R2$.

3.3.2.1 DF Elementar

Definição

Seja $R(X, Y, Z, \dots)$ com X, Y, Z conjuntos de atributos,

a DF $X \rightarrow Y$ é **elementar** se $\forall X' \subset X: X' \not\rightarrow Y$

3.3.2.2 DF Parcial

Definição

Se $X \rightarrow Y$ e $X' \rightarrow Y$ (com $X' \subset X$) diz-se que

*$X \rightarrow Y$ é uma **dependência funcional parcial**.*

3.3.2.3 Segunda Forma Normal (2FN)

Seja $R(A_1, A_2, \dots, A_n)$.

R está na 2FN sse $\forall A_i \notin \text{chave}(s), \forall_X \text{ chave}$, se verifica que

$X \rightarrow A_i$ é elementar.

De outra forma:

- Uma relação está em 2FN se está em 1FN e os atributos que não são chave dependem da totalidade da chave.

Nota:

*Um atributo pertencente a uma chave diz-se um atributo **primo**.*

Exemplo:

*Linha_nota_enc (N_nota_enc, Cod_produto, Desc_produto, Preço_prod,
Quantidade)*

As dependências funcionais,

$N_nota_enc, Cod_produto \rightarrow Desc_produto$

$N_nota_enc, Cod_produto \rightarrow Preço_produto$

não são elementares, porque

$$Cod_produto \rightarrow Desc_produto \quad (1)$$
$$Cod_produto \rightarrow Preço_produto \quad (2)$$

A relação *Linha_nota_enc* vai dar origem a:

Linha_Nota_Enc (*N_nota_enc*, *Cod_produto*, *Quantidade*)

Produto (*Cod_produto*, *Desc_produto*, *Preço_produto*)

A DF $Cod_produto \rightarrow Desc_produto, Preço_produto$ (união de (1) e (2)) garante que a decomposição é válida.

3.3.2.4 Casos especiais de relações na 2FN:

- . Se todos os atributos de uma relação são primos.
- . A chave da relação consiste num único atributo.

Nota: A definição de 2FN não “proíbe” a existência de DF parciais entre atributos primos.

3.3.2.5 Decomposição em 2FN

Toda a relação R que não esteja na 2FN pode ser decomposta num conjunto de projeções que estão na 2FN.

Dem.

Suponhamos que $R(A_1, A_2, \dots, A_n)$ não está na 2FN.

Então existem subconjuntos disjuntos de $\{A_1, A_2, \dots, A_n\}$, X e Y , tais que:

- *Y não tem atributos chave;*
- *X é chave;*
- *Existe a DF parcial $X \rightarrow Y$.*

Seja Z o conjunto dos atributos que não pertencem nem a X nem a Y .

R pode ser representada por $R(X, Y, Z)$.

Se $X \rightarrow Y$ é uma DF parcial, então X pode ser representado por $X'X''$ onde $X' \rightarrow Y$ é uma DF elementar.

Num primeiro passo decompõe-se $R(XYZ)$ em $\pi_{X,Y}(R)$ e $\pi_{X,Z}(R)$.

- *$\pi_{X,Y}(R)$ está na 2FN porque:*

. $X' \rightarrow Y$ é elementar

. Y contém todos os atributos não primos

- *Se $\pi_{X,Z}(R)$ não está em 2FN aplicamos novamente o procedimento anterior.*

—

Resumindo, para vermos se uma relação está na 2FN:

1 – Identificamos a chave da tabela. Se a chave for apenas um atributo, ou for constituída por todos os atributos da relação, então podemos concluir que está na 2FN.

2 – Se a chave for composta (tiver mais do que um atributo) verificamos se há atributos que não são chave e que dependem apenas de parte da chave. Se não houver, então está na 2FN.

Se houver, então decompor de acordo com o procedimento anterior.

3.3.3 Terceira Forma Normal (3FN)

Seja $E(\underline{N_emp}, Dep, Cidade)$, onde:

$$N_emp \rightarrow Dep$$

$$Dep \nrightarrow N_emp$$

$$Dep \rightarrow Cidade$$

$$Cidade \nrightarrow Dep$$

E

N_emp	Dep	Cidade
a	A	X
b	A	X
c	A	X

d	B	Y
e	B	Y
f	C	X
g	C	X

Anomalias:

Inserção: não podemos inserir um departamento se não tem empregados

Eliminação: Se eliminamos o último empregado de um departamento perdemos a informação do departamento

Modificação: O atributo cidade é repetido para cada empregado de um dado departamento; uma alteração da localização de um departamento implica alterar a localização de todos os empregados desse departamento.

$E(\underline{N_emp}, Dep, Cidade)$ está em 2FN!

Se substituirmos E por:

$$\text{Empregado} = \pi_{N_emp, Dep}(E)$$

e

$$\text{Departamento} = \pi_{Dep, Cidade}(E)$$

desaparecem as anomalias.

A DF $Dep \rightarrow Cidade$ garante que $E = \text{Empregado} \bowtie \text{Departamento}$.

Ficamos com o esquema relacional:

Empregado (*N_emp*, *Dep*)

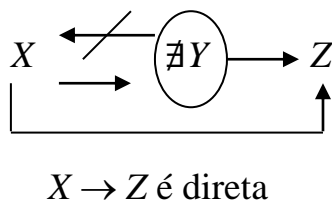
Departamento (*Dep*, *Cidade*)

3.3.3.1 DF Direta

Definição

Seja $R(X, Y, Z, \dots)$ $X \rightarrow Z$ é **direta** sse

$\nexists Y \in R: Y \nrightarrow X, X \rightarrow Y$ e $Y \rightarrow Z$ (não trivial).



3.3.3.2 Terceira Forma Normal (3FN)

Seja $R(A_1, A_2, \dots, A_n)$.

R está na 3ª Forma Normal sse

$\forall A_i \notin \text{chave}(s), \forall X \text{ chave, se verifica que } X \rightarrow A_i \text{ é direta.}$

De outra forma:

- Uma relação está na 3FN se está na 2FN e nenhum dos atributos não chave depende de outro também não chave!

3.3.3.3 3FN e 2FN

Se uma relação está na 3FN então está também na 2FN

Resolução:

Suponhamos que R está na 3FN mas que tem um atributo A_i (não chave) que depende parcialmente de um conjunto de atributos X , com X chave de R (ou seja, R não está na 2FN).

Para algum $X' \subset X$, $X \rightarrow X' \rightarrow A_i$.

Logo, $X \rightarrow A_i$ não é direta.

Portanto, R não estaria na 3FN — Contradição.

3.3.3.4 Decomposição em 3FN

Toda a relação R que não esteja na 3FN pode ser decomposta num conjunto de projeções que estão na 3FN.

Dem.

R não está em 3FN.

Portanto, existe $X \rightarrow Y$, $Y \nrightarrow X$ e $Y \rightarrow Z$ (não trivial), com X chave e Z não-chave (isto é, existe $X \rightarrow Z$ não direta)

Decompor em R em $\pi_{Y,Z}(R)$ e $\pi_{X,Y,W}(R)$, onde W tem todos os atributos que não são de X , nem de Y nem de Z .

Repetir o processo se necessário.

Exemplo 1:

Nota_de_enc (*N_not_a_enc*, *Cod_cliente*, *Nome_cliente*, *Morada_cliente*)

Cod_cliente \rightarrow *Nome_cliente*

Cod_cliente \rightarrow *Morada_cliente*

por união, *Cod_cliente* \rightarrow *Nome_cliente*, *Morada_cliente*

(logo a DF *N_not_a_enc* \rightarrow *Nome_cliente*, *Morada_cliente* (não é directa)

A relação não está na 3FN.

Decomposição:

Cliente (*Cod_cliente*, *Nome_cliente*, *Morada_cliente*)

Nota_enc (*N_not_a_enc*, *Cod_cliente*)

Exemplo 2:

Cliente (*Código*, *Nome*, *Morada*, *Cod_postal*)

onde, $\left. \begin{array}{l} \textit{Código} \rightarrow \textit{Nome} \\ \textit{Nome} \rightarrow \textit{Código} \\ \textit{Código} \rightarrow \textit{Morada} \end{array} \right\} \Rightarrow \text{(por transitividade) } \textit{Nome} \rightarrow \textit{Morada}$

Código \rightarrow *Morada* - Esta dependência funcional é directa?

A relação está na 3ª forma normal?

Resposta: _____

Exercício: Normalize em 3FN o esquema relacional abaixo.

Clientes (*N_cliente*, (*Endereços_para_remissa*) *, *Saldo*,
Limite_de_crédito, *Desconto*)

* - grupo repetitivo

Peças (*N_peça*, *Cod_armazém*, *Qtd_stock_armazém*,
Min_stock_armazém, *Desc_peça*, *Cor*)

- Uma peça pode existir em vários armazéns.

Encomendas (*N_enc*, *Cod_cliente*, *Endereço_remissa*, *Data*,
(*N_peça*, *Qtd_pedida*, *Qtd_enviada*)*)

3.3.4 Forma Normal de Boyce-Codd (FNBC)

Seja *R* (*Cidade*, *Endereço*, *Cod_postal*) com

Cidade, *Endereço* → *Cod_postal*

Cod_postal → *Cidade*

Chaves:

Cidade, *Endereço*

Endereço, *Cod_postal*

Cidade	Endereço	Cod_postal
c1	e1	p1
c1	e2	p2
c1	e3	p2
c2	e4	p3

A relação está na 3FN mas existem algumas anomalias:

Inserção: Não podemos inserir o código postal de uma cidade se não especificarmos o endereço.

Eliminação: Ao eliminarmos o último endereço de um dado código postal perdemos a cidade correspondente.

Modificação: Se é alterado o código postal de uma cidade é necessário alterar o código postal de todos os endereços com esse código postal.

3.3.4.1 Definição (FNBC)

Seja $R (A_1, A_2, \dots, A_n)$. R está na **Forma Normal de Boyce-Codd** sse para qualquer DF $X \rightarrow Y$ (não trivial), X e Y conjuntos de atributos de R , X é chave candidata de R .

De outra forma:

- Uma relação está na FNBC se e só se todo o determinante da relação for uma chave candidata.

Voltando ao exemplo anterior:

Chaves:

Cidade, Endereço

Endereço, Cod_postal

e existe a DF *Cod_postal* \rightarrow *Cidade*

Cod_postal não é chave candidata logo a relação não está na FNBC.

Decomposição:

R1 (Cod_postal, Cidade)

R2 (Endereço, Cod_postal)

Observação: A FNBC só é diferente da 3FN se a relação tem mais do que uma chave.

Exemplo 1:

E (Emp, Dep, Cidade)

Emp \rightarrow *Dep*

Dep \nrightarrow *Emp*

Dep \rightarrow *Cidade*

Cidade \nrightarrow *Dep*

Não está na 3FN (porque a DF *Emp* \rightarrow *Cidade* não é direta)

Não está na FNBC (porque na DF *Dep* \rightarrow *Cidade* o determinante não é chave candidata).

Decomposição:

Duas decomposições sem perda de informação. Qual escolher?

(1)

$E1 (\underline{Emp}, Dep)$

$E2 (\underline{Emp}, Cidade)$

$Emp \rightarrow Dep$
 $Emp \rightarrow Cidade$

ou

(2)

$E1 (\underline{Emp}, Dep)$

$E2 (\underline{Dep}, Cidade)$

$Emp \rightarrow Dep$
 $Dep \rightarrow Cidade$

Em (1) perdemos a DF $Dep \rightarrow Cidade$.

Em (2) preservamos as DFs ($Emp \rightarrow Cidade$ obtém-se por transitividade). A decomposição (2) é, portanto, a decomposição correta.

Exemplo 2:

$Nota_enc(\underline{N_nota_enc}, Cod_cliente, Nome_Cliente, Morada_cliente)$

$Cod_cliente \rightarrow Nome_cliente$

$Cod_cliente \rightarrow Morada_cliente$

Não está na 3FN.

Não está na FNBC

(porque na DF $Cod_cliente \rightarrow Nome_cliente, Morada_cliente$,

$Cod_cliente$ não é chave candidata)

Decomposição:

$NE1 (\underline{N_nota_enc}, Cod_cliente)$

$NE2 (\underline{Cod_cliente}, Nome_cliente, Morada_cliente)$

3FN ✓
FNBC ✓

Exemplo 3: Relação com duas chaves candidatas não sobrepostas.

$F(N_{\text{forn}}, Nome_{\text{forn}}, Cidade, Tipo)$

$N_{\text{forn}} \rightarrow Nome_{\text{forn}}$

$Nome_{\text{forn}} \rightarrow N_{\text{forn}}$

$N_{\text{forn}} \rightarrow Cidade, Tipo$

$Nome_{\text{forn}} \rightarrow Cidade, Tipo$

Chaves candidatas:

N_{forn}

$Nome_{\text{forn}}$

3 FN? _____

FNBC? _____

Exemplo 4: Relação com duas Chaves candidatas sobrepostas.

$F(N_{\text{forn}}, Nome_{\text{forn}}, N_{\text{peça}}, Qtd)$

$N_{\text{forn}} \rightarrow Nome_{\text{forn}}$

$Nome_{\text{forn}} \rightarrow N_{\text{forn}}$

$N_{\text{forn}}, N_{\text{peça}} \rightarrow Qtd$

Chaves candidatas:

$N_{\text{forn}}, N_{\text{peça}}$

$Nome_{\text{forn}}, N_{\text{peça}}$

3FN? *sim*

FNBC? *não, porque na DF $N_{\text{forn}} \rightarrow Nome_{\text{forn}}$,*

N_{forn} não é chave candidata.

Decompôr em:

$F1(N_{\text{forn}}, Nome_{\text{forn}})$

$F2(N_{\text{forn}}, N_{\text{peça}}, Qtd)$

Ambas estão na FNBC.

3.3.4.2 FNBC e 3FN

Se uma relação está na FNBC então está na 3FN

Dem.

Seja $R(A_1, A_2, \dots, A_n)$ na FNBC.

Suponhamos que existe uma DF não direta $X \rightarrow A_i$ tal que $X \rightarrow Y$, $Y \nrightarrow X$ e

$Y \rightarrow A_i$ para algum A_i não primo e X chave. (Isto é, R não está na 3FN).

Mas, se existe $Y \rightarrow A_i$ (não trivial) então Y é chave candidata de R (porque por hipótese R está na FNBC) e, portanto, $Y \rightarrow X$ – Contradição.

3.3.4.3 Decomposição em FNBC.

Se uma relação não está na FNBC pode ser decomposta num conjunto de projeções.

Dem.

Seja $R(X, Y, Z)$ uma relação que não está na FNBC, onde X , Y e Z são conjuntos de atributos tais que $X \rightarrow Y$ (não trivial) e $X \nrightarrow Z$ (logo X não é chave candidata de R).

- Substituir $R(X, Y, Z)$ por $\pi_{X, Y}(R)$ e $\pi_{X, Z}(R)$.
- Se necessário repetir o processo.

Exercício:

Decomponha em 3FN e em FNBC.

Proprietário (*N_carro*, *Marca*, *Tipo*, *Cor*, *BI*, *Nome*, *Data*, *Preço*),

onde:

$N_carro \rightarrow Cor, Tipo$

$Tipo \rightarrow Marca, Preço$

$BI \rightarrow Nome$

$N_carro, Nome \rightarrow Data$

3.3.5 Preservação de dependências¹⁴

As dependências funcionais representam restrições de integridade, e, portanto, devem ser mantidas nas relações resultantes da decomposição.

Intuitivamente, uma decomposição preserva as dependências se nos permite garantir que todas as DFs são respeitadas através da inspeção de uma só relação em cada inserção ou alteração de um tuplo. (Note-se que as eliminações não causam violação de DFs.) Uma definição mais rigorosa exige a introdução do conceito de projeção de DFs.

Seja R um esquema de relação que é decomposto em dois esquemas de relação com conjuntos de atributos X e Y , e seja F um conjunto de DFs em R . A **projeção de F sobre X** é o conjunto de DFs no fecho F^+ que contém apenas atributos em X . A projeção de F sobre X representa-se por F_X .

Uma DF $U \rightarrow V$ em F^+ só pertence a F_X se todos os atributos em U e V pertencerem a X .

¹⁴ (Ramakrishnan & Gehrke, 2003), pág. 621.

A decomposição da relação R com DFs F em esquemas de relação com conjuntos de atributos X e Y , **preserva as dependências** se $(F_X \cup F_Y)^+ = F^+$. Isto é, se tomarmos as dependências em F_X e F_Y e calcularmos o fecho da sua união, obtemos todas as dependências do fecho de F . Portanto, só precisamos de garantir as dependências em F_X e F_Y ; todas as DFs de F^+ ficam garantidamente satisfeitas. Para garantir F_X só é necessário examinar a relação X (nas inserções nesta relação). Para garantir F_Y só é necessário examinar a relação Y .

Exemplo:

Sejam $R(A, B, C)$ um esquema de relação e F o conjunto das DFs de R , com $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$.

Suponha que R é decomposta em AB e BC . Será que esta decomposição preserva as dependências?

$A \rightarrow B$ está em F_{AB} .

$B \rightarrow C$ está em F_{BC} .

E acerca de $C \rightarrow A$? Esta dependência não é implicada pelas dependências listadas (até agora) em F_{AB} e F_{BC} .

F^+ contém todas as dependências de F e ainda contém $A \rightarrow C$, $B \rightarrow A$, e $C \rightarrow B$. Por conseguinte, F_{AB} contém $B \rightarrow A$, e F_{BC} contém $C \rightarrow B$. Portanto, $F_{AB} \cup F_{BC}$ contém $A \rightarrow C$, $B \rightarrow C$, $B \rightarrow A$ e $C \rightarrow B$. O fecho das dependências em F_{AB} e em F_{BC} inclui $C \rightarrow A$ (devido a $C \rightarrow B$, e $B \rightarrow A$, por transitividade). Donde, a decomposição preserva a dependência $C \rightarrow A$.

Uma aplicação direta da definição dá-nos um algoritmo direto para testar quando é que uma decomposição preserva as dependências funcionais.

A desenvolver... (em futuras versões do documento)

- Seja R uma relação.

R pode ser decomposta num conjunto de projeções que estão na 3FN e que preservam as dependências.

- Seja R uma relação.

Não é garantido que se possa decompor R num conjunto de projeções na FNBC preservando as dependências.

4 Modelo Entidade–Associação

4.1 Introdução

Os modelos de dados facilitam a interação entre projetistas, programadores e utilizadores finais. Um modelo bem projetado promove uma melhor compreensão da organização. Ou seja, os modelos de dados são, na verdade, ferramentas de comunicação, por excelência.

Modelo de Dados

- . Visão dos dados em vez de visão das aplicações
- . Eliminação de redundâncias
- . Partilha de dados pelas aplicações

Construir um modelo de dados é:

- Identificar, Analisar e Registrar a política da organização acerca dos dados.

A definição de um modelo de dados é feita a três níveis:

- Conceptual

Representação fiel da realidade, sem atender a quaisquer constrangimentos impostos pelo modelo informático.
- Lógico

Adaptação do modelo conceptual a um modelo de dados específico (ex. modelo relacional), mas independente de qualquer SGBD ou outras considerações físicas.

- Físico

Adaptação do modelo lógico às características do sistema informático.

As técnicas de modelação dividem-se em dois grupos:

➤ Do particular para o geral (*Bottom-up*)

Parte da identificação dos níveis mais elementares (atributos) de informação e agrupa-os usando as relações de interdependência entre eles (dependências funcionais). Esta é a abordagem da Teoria da Normalização (Codd, 1970)

Adequada para pequenos projetos (6-8 tabelas).

➤ Do geral para o particular (*Top-down*)

Parte dos grandes objetos de informação (entidades) identificando as suas inter-relações.

1º - Selecionar entidades e associações entre elas que tenham interesse para a organização.

2º - Especificar os atributos para cada entidade e associação.

Adequado para grandes projetos.

Esta é a abordagem do Modelo Entidade–Associação.

4.2 Modelo Entidade-Associação

O surgimento, em 1970, do modelo relacional trouxe simplicidade e flexibilidade ao mundo das bases de dados, quando comparado com os modelos hierárquico e rede. Contudo, continuava a faltar uma ferramenta que tornasse mais efetivo o projeto de bases de dados. Na verdade, faltava uma ferramenta que permitisse aos projetistas **ver em vez de ler**, pois é mais fácil examinar estruturas graficamente do que ler as suas descrições textuais.

Em 1976, Peter Chen introduziu o *modelo entidade/associação* (*entity/relationship diagram*) que permitia representar graficamente as entidades e as suas associações. O modelo entidade/associação rapidamente ganhou popularidade devido à sua complementaridade com o modelo de dados relacional¹⁵.

Note-se que o modelo Entidade/Associação também pode ser usado por outros modelos de dados.

O Modelo Entidade/Associação é especificado a dois níveis:

Gráfico

Diagrama entidade–associação (DEA)

Descritivo

Especificação de cada componente do modelo

¹⁵ (Coronel & Morris, 2016), pág. 46.

Componentes do Modelo Entidade/Associação:

Entidade — Qualquer coisa (objeto ou conceito) com interesse para a organização, a respeito da qual é guardada informação, e que possa ser identificada de maneira inequívoca.

Exemplos: Funcionário, Departamento, Contrato.

Para cada entidade é necessário conhecer quais as propriedades que são relevantes para o sistema.

Atributo — Atributo é qualquer propriedade de uma entidade. Um atributo é um elemento atômico (indivisível) de informação.

Exemplos: N° de empregado, Nome, ...

Associação — As associações relacionam duas ou mais entidades.

Uma associação relaciona:

- duas entidades entre si (associação binária)
- várias entidades entre si (associação complexa)
- uma entidade com ela própria (associação unária)

Por vezes uma associação limita-se a relacionar entidades entre si.

Há situações em que as associações possuem propriedades próprias.

Distinguir entidades de associações:

- . Substantivos - para fazer referências a entidades
- . Verbos – para fazer referência a associações

Tipos de atributos:

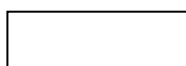
- . Identificadores (chaves)
- . Descritores

Diagrama Entidade/Associação:

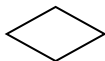
Um diagrama E/A é um grafo representando entidades e associações. Os nós do grafo são representados por formas especiais, de acordo com o seu tipo.

Na notação de Chen (1976),

- entidades são representadas por retângulos;
- as associações são representadas por losangos.



Entidade



Associação



Associação com atributos

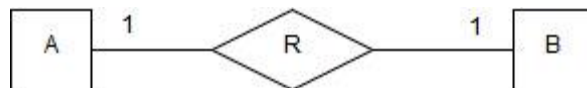
4.3 Propriedades das associações

4.3.1 Grau de uma associação.

As associações distinguem-se pelo seu grau:

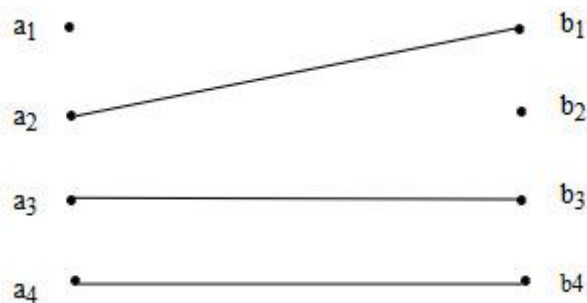
- Associação $1:1$ (um para um)
- Associação $1:N$ (um para vários)
- Associação $M:N$ (vários para vários)

4.3.1.1 Associação 1:1



- A cada ocorrência da entidade *A* está associada apenas uma ocorrência da entidade *B* (ou nenhuma).
- A cada ocorrência da entidade *B* está associada apenas uma ocorrência da entidade *A* (ou nenhuma).

Diagrama de ocorrências:

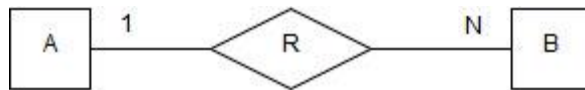


Exemplo:

Seja um curso em que cada módulo é assegurado por um monitor e cada monitor assegura apenas um módulo

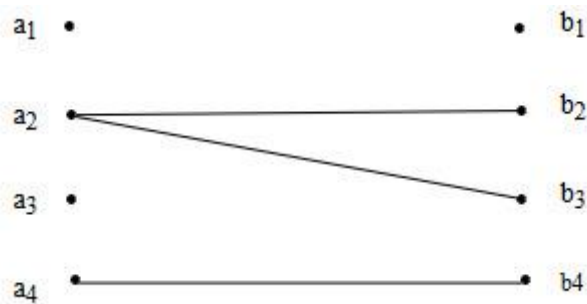


4.3.1.2 Associação 1:N



- A cada ocorrência da entidade *A* está associada uma, várias ou nenhuma ocorrência da entidade *B*.
- A cada ocorrência da entidade *B* está associada apenas uma ocorrência da entidade *A* (ou nenhuma).

Diagrama de ocorrências:



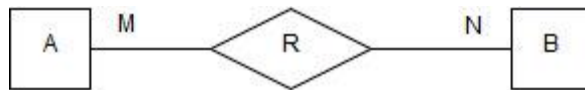
Exemplo:

Um departamento tem atribuídos vários empregados (eventualmente só um, ou mesmo nenhum).

Um empregado está atribuído a apenas um departamento (ou nenhum).

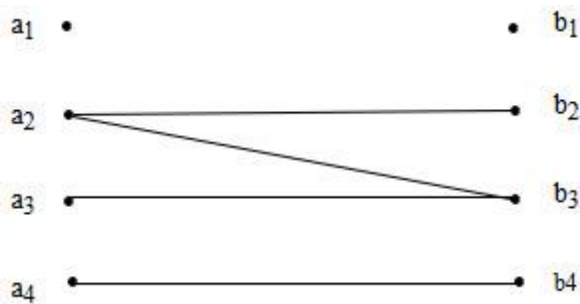


4.3.1.3 Associação $M:N$



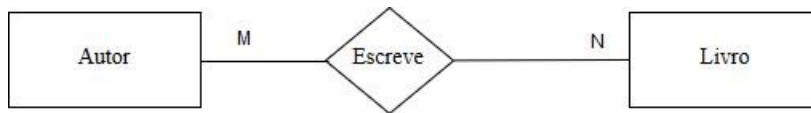
- A cada ocorrência da entidade A está associada uma, várias ou nenhuma ocorrência da entidade B .
- A cada ocorrência da entidade B está associada uma, várias ou nenhuma ocorrência da entidade A .

Diagrama de ocorrências:



Exemplo:

Um autor pode escrever vários livros, e livro pode ser escrito por vários autores.



Exercícios:

1. Um modelo conceptual de dados deverá conter os atributos *endereço_propriedade*, *n_de_quartos*, *valor_aluguer* e *nome_proprietário*.

Um proprietário não tem necessariamente de ocupar uma casa que é a sua.

A estrutura de dados deverá permitir obter:

- Quem é o dono de uma dada propriedade?
- Que propriedade um proprietário ocupa?

Sugira dois tipos de entidades e duas possíveis associações entre elas.

Desenhe um diagrama entidade/associação.

2. Para cada par de restrições abaixo, identifique dois tipos de entidades e um tipo de associação. Indique o grau da associação para cada caso.

a) Um departamento emprega várias pessoas.

Uma pessoa trabalha para quando muito um departamento.

b) Um gestor chefia no máximo um departamento.

Um departamento é chefiado quando muito por um gestor.

c) Uma equipa consiste em vários jogadores.

Um jogador joga para uma só equipa.

d) Um professor lociona no máximo um curso.

Um curso é locionado por um só professor.

e) Uma nota de encomenda pode ter vários produtos.

Um produto pode aparecer em várias notas de encomenda.

f) Um cliente pode receber várias faturas.

Uma fatura é de um só cliente.

3. Numa clínica médica, cada médico tem vários doentes, mas um doente só pode estar registado num médico de cada vez. Supondo que só se incluem os registos de doentes atuais, qual é o grau da associação *Registado* entre as entidade *Doente* e *Médico*.

Desenhe um diagrama entidade/associação.

4. Qual a resposta à questão 3 se o modelo for alterado para incluir um histórico de todos os registos da cada doente.

5. Se na questão 3 um paciente pudesse registar-se simultaneamente em vários médicos qual seria o grau da associação?

6. Na questão 2, o que aconteceria à associação 1:1 entre *Chefe* e *Departamento* se fosse necessário ter um histórico dos registos.

4.3.2 Tipo de participação

Uma entidade pode participar numa associação de duas formas:

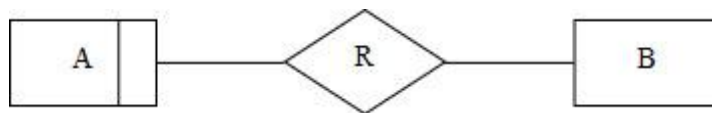
- **Obrigatória**

Todas as ocorrências dessa entidade têm de estar associadas a alguma ocorrência da outra entidade que participa na associação.

- **Não obrigatória**

A entidade pode ter ocorrências não associadas a qualquer ocorrência da outra entidade que participa na associação

Representa-se,



- Entidade *A*: obrigatória.
- Entidade *B*: não-obrigatória.

Exemplos:

1)



- Um departamento tem de ter pelo menos um empregado
- Um empregado tem de pertencer a um departamento

2)



- Um departamento pode não ter empregados
- Um empregado tem de pertencer a um departamento

3)



- Um departamento tem de ter pelo menos um empregado
- Um empregado pode não pertencer a um departamento

4)



- Um departamento pode não ter empregados
- Um empregado pode não pertencer a um departamento

Exercícios:

1. Decida plausíveis tipos de participação (obrigatória/não-obrigatória):

<u>Entidades</u>	<u>Associação</u>
a) Casa, Pessoa	É_proprietário
b) Casa, Inquilino	Habita
c) Encomenda, LinhaEncomenda	Contém
d) Zona_Vendas, Cliente	Possui
e) ClienteBanco, ContaCliente	Possui
f) Empregado, Habitação	Tem

2. Que dificuldade prática pode ocorrer quando se inserem os dados relativos a um departamento e a um empregado numa base de dados em que:

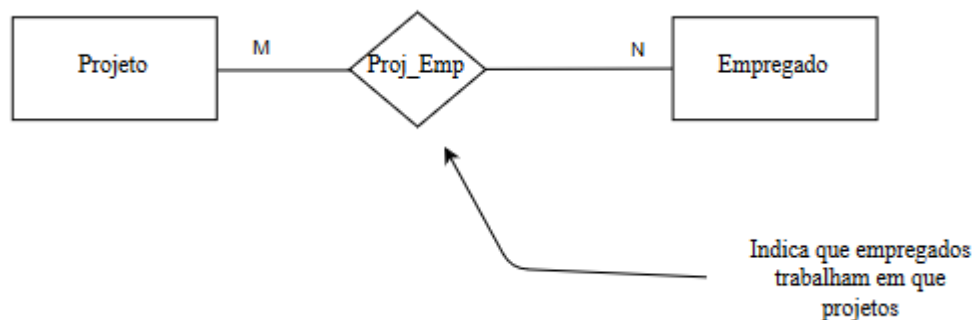
- Um departamento tem de ter pelo menos um empregado
- Um empregado tem de pertencer a um departamento

4.4 Decomposição de Associações $M:N$

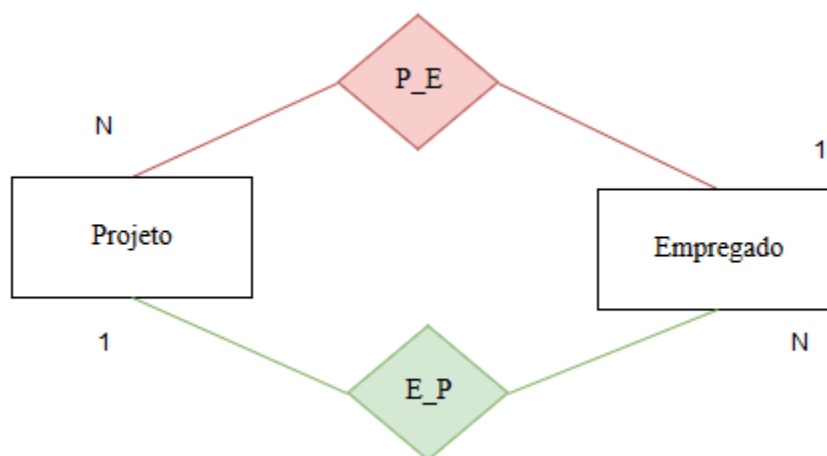
Nas fases iniciais do processo de modelação podem surgir associações muitos-para-muitos ($M:N$). No entanto, na fase final da modelação de dados não devem aparecer relações $M:N$ pois não são diretamente suportadas pelos SGBDs, levando à criação de uma nova **entidade associativa** ou de intersecção.

Qualquer associação $M:N$ entre dois tipos de entidades pode ser decomposta em duas associações $1:N$.

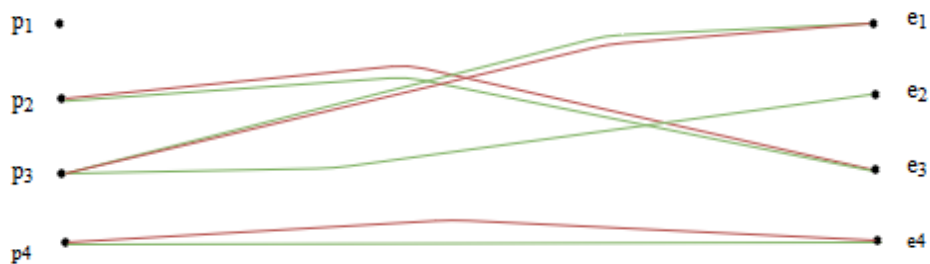
Seja a associação:



Note-se que este DEA é diferente do seguinte:

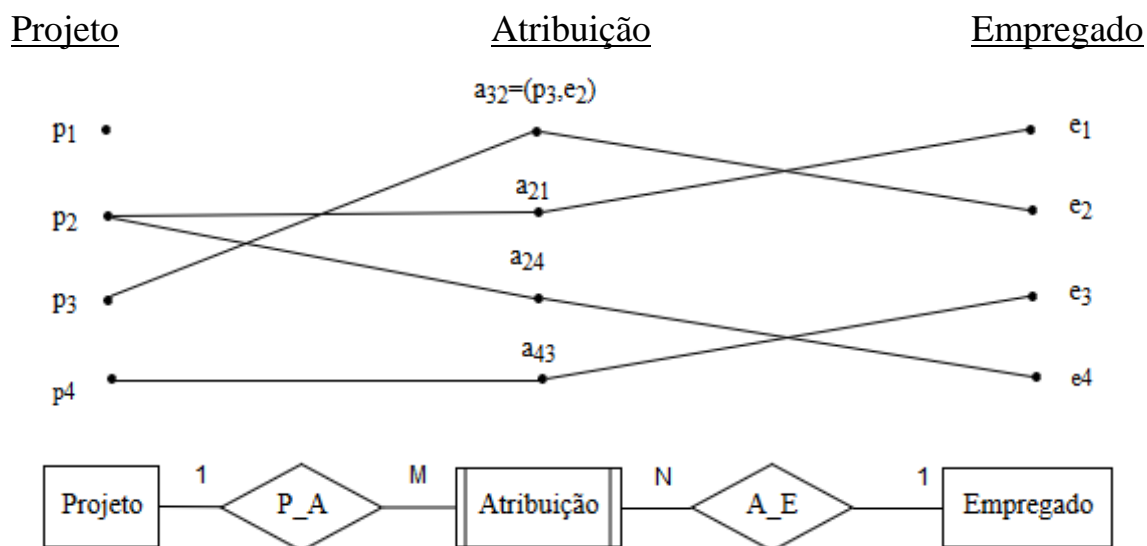


Repare-se no diagrama de ocorrências:



Tem-se duas associações entre as mesmas entidades, a representar papéis distintos.

Na associação inicial, cada ocorrência da associação corresponde à atribuição de um projeto a um empregado, realçando assim o surgir de uma nova entidade (entidade escondida: *Atribuição*):



Exercícios:

1. A tabela seguinte mostra que fornecedores fornecem que peças:

CodForn	CodePeça
F1	P15
F1	P29
F1	P32
F2	P12
F2	P15
F3	P12
F3	P32

a) Desenhar um diagrama E-A mostrando uma associação entre as entidades Fornecedor e Peça

b) Decompor o diagrama de forma a que apenas contenha associações 1:N.

2. Uma escola possui vários cursos (arte, dança, música, história, ...).

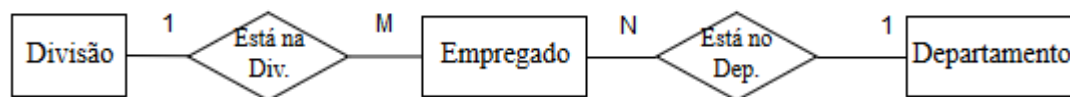
Cada curso pode ter vários professores e um professor pode dar vários cursos. Um determinado curso utiliza sempre a mesma sala. Mas cursos diferentes podem utilizar a mesma sala.

a) Desenhar um diagrama E-A mostrando as entidades *Professor*, *Curso* e *Sala* e as associações *Prof_Curso* e *Sala_Curso*.

b) Redesenhe o diagrama decompondo associações *M:N* em associações *1:N*

3. O diagrama abaixo é semelhante à decomposição de uma associação de *M:N* entre *Divisão* e *Departamento*.

Poderá a associação *Divisão_Contém_Departamento* ser *1:N* ou terá de ser necessariamente *M:N*?



4. As afirmações seguintes são verdadeiras ou falsas?

a) Qualquer associação *M:N* pode ser decomposta em duas associações *1:N*.

b) A estrutura *X (1:N) Y (N:1) Z* significa que tem que existir uma associação *M:N* entre *X* e *Z*.

5. Numa empresa cada empregado tem no máximo uma habilitação, mas uma dada habilitação pode ser possuída por vários empregados.

Um empregado está habilitado a operar determinado tipo de máquina se tiver uma de entre várias habilitações, mas cada habilitação está associada com a operação de um único tipo de máquina.

Possuir uma dada habilitação permite ao empregado manter vários tipos de máquinas apesar de a manutenção de um tipo de máquina requerer uma habilitação específica.

- a) Desenhe o diagrama E-A usando as seguintes entidades e associações:

<u>Entidade</u>	<u>Associação</u>	<u>Entidade</u>
Habilitação	Possuída_por	Empregado
Tipo_Máquina	Necessita_para_Operação	Habilitação
Habilitação	Permite_Manutenção	Tipo_Máquina

- b) Redesenhe o diagrama com as associações

Necessita_para_Operação e *Permite_Manutenção* combinados numa só associação *M:N, Operação_ou_Manutenção*.

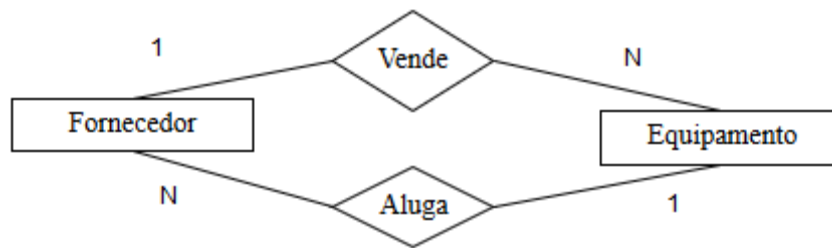
- c) Redesenhe o diagrama anterior com a associação

Operação_ou_Manutenção decomposta em associações *1:N*.

- d) Porque não é possível inverter o processo, isto é, dado o diagrama c) convertê-lo em a)?

6. Fornecedores vendem e alugam equipamento como está ilustrado abaixo.

Um fornecedor pode alugar equipamento vendido por outros fornecedores.



Vendas	Forn	Equip	Alugueres	Forn	Equip
	F1	E1		F1	E1
	F1	E2		F1	E3
	F2	E3		F2	E4
	F2	E4			

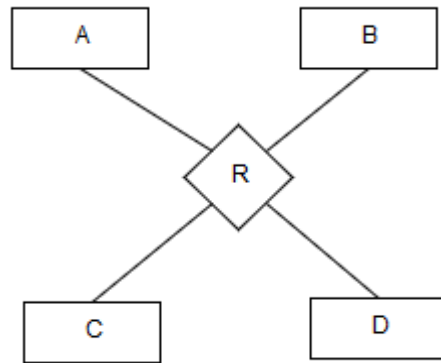
- a) Crie uma tabela de ocorrências que mostre que equipamento cada fornecedor vende **ou** aluga.

A combinação das duas associações $1:N$ entre *Fornecedor* e *Equipamento*, numa só associação, deu origem a uma associação $M:N$ ou $1:N$?

- b) Sugira uma alteração às regras da organização que sem alteração do diagrama torna a associação *Vende_ou_aluga* $1:N$.

4.5 Associações Complexas

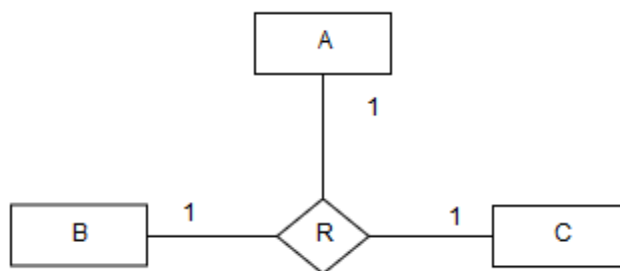
Relacionam entre si mais do que duas entidades.



Devem ser usadas apenas quando o conceito inerente à associação não pode ser representado por um conjunto de associações binárias.

Nota: para se estabelecer o grau da associação, considera-se uma ocorrência de cada uma das $N-1$ entidades envolvidas e vê-se como se conjugam com a entidade que ficou livre.

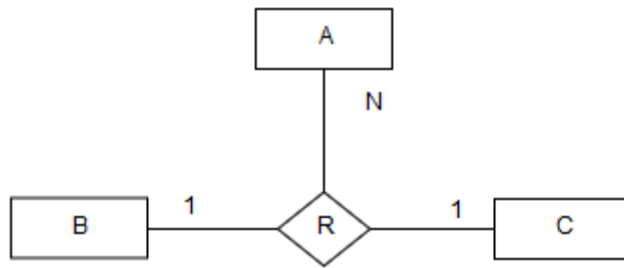
4.5.1 Grau 1:1:1



A cada par de ocorrências das entidades *B* e *C* está associada apenas uma ocorrência de *A* ou nenhuma.

Análogo para *B* e *C*.

4.5.2 Grau 1:1:N



. A cada par de ocorrências das entidades *B* e *C* está associada uma, várias ou nenhuma ocorrência de *A*.

. A cada par de ocorrências de *A* e *C* está associada apenas uma ocorrência de *B* (ou nenhuma).

. A cada par de ocorrências *A* e *B* está associada apenas uma ocorrência de *C* (ou nenhuma).

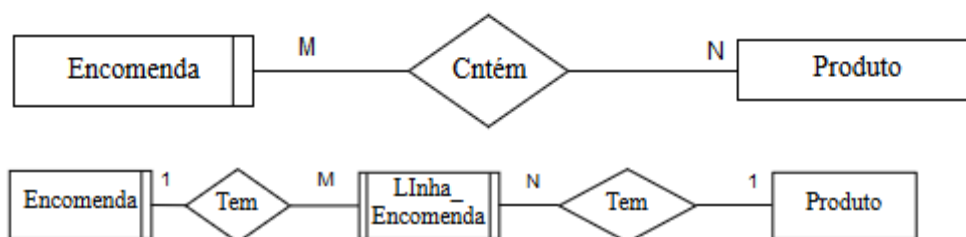
Análogo para,

Grau $1:M:N$

Grau $M:N:P$

Vimos que:

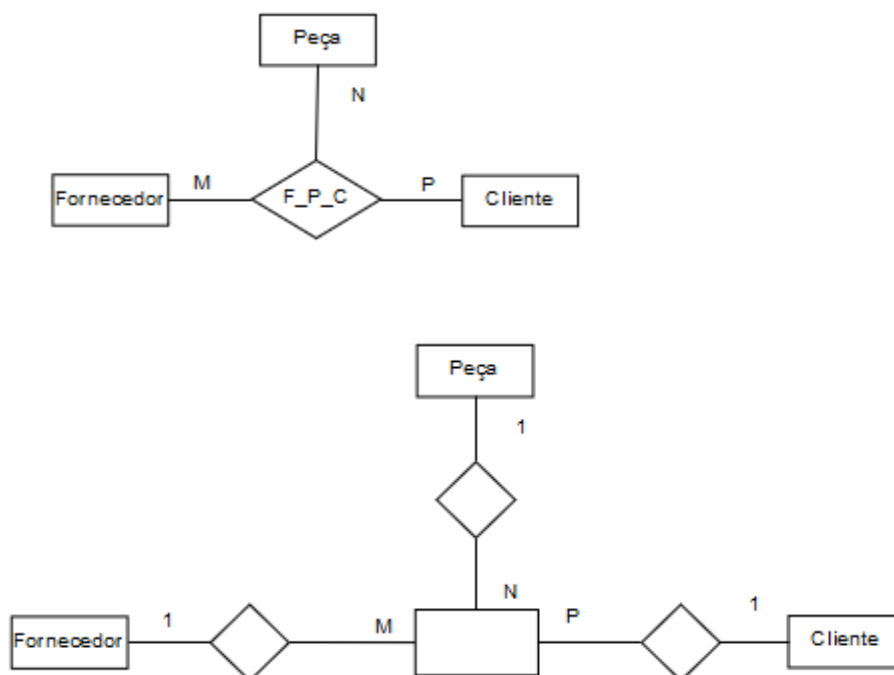
As associações $M:N$ devem ser substituídas por um par de associações de grau $1:N$.



A decomposição permite:

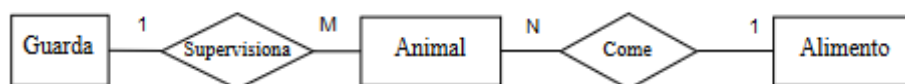
- Ressaltar a existência de entidades não identificadas no início;
- (Facilita) a análise do diagrama em termos de consistência;
- Dar ao modelo a forma adequada para passos subsequentes do método.

Associações complexas devem ser decompostas em associações binárias:



4.6 Situações Ambíguas

A estrutura,



Pode representar:

- a) Que guarda supervisiona que animal?
- b) Que alimentos um animal come?

- c) Que comida um guarda supervisiona?
- d) Que comida um guarda come?
- e) Que animal gosta de que comida?
- f) Que animal é comido por que guarda?

Nota: não há transitividade! As leituras aceitas são aquelas que estão diretamente representadas. Todas as outras são abusivas e erradas.

Problemas com os modelos entidade/associação:

Os problemas normalmente ocorrem devido a uma interpretação incorreta do significado das associações. De entre as armadilhas de ligação, duas tem especial interesse: ausência de informação (abismo: *chasm trap*) e a ambiguidade de informação (laço: *fan trap*).

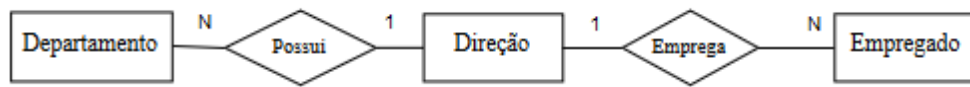
Em geral, para identificar as armadilhas de ligação, deve-se assegurar que o significado da associação é totalmente compreendido e claramente definido. Se não compreendermos as associações, podemos criar um modelo que não é uma verdadeira representação do mundo real¹⁶.

Fan trap - Where a model represents a relationship between entity types, but the pathway between certain entity occurrences is ambiguous.

Uma *fan trap* pode ocorrer quando duas ou mais associações 1:N se ligam à mesma entidade.

¹⁶(Connolly & Begg, 2015), pág. 426.

Exemplo 1:



Permite responder às questões:

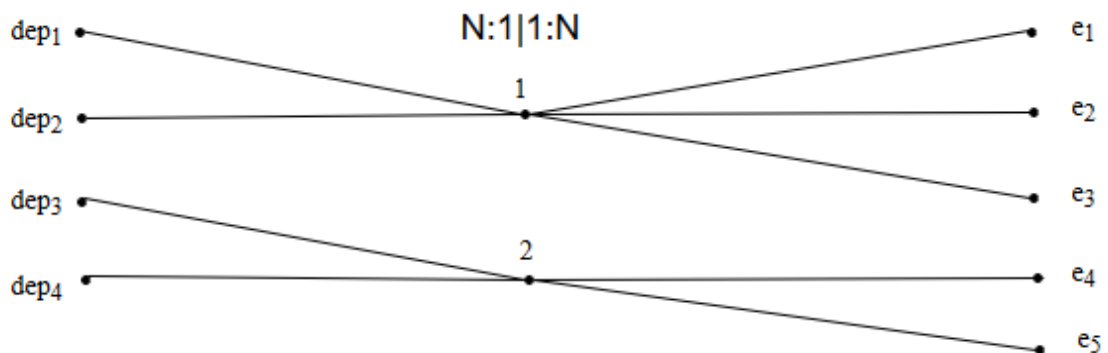
- Quais os empregados de uma direção?
- Quais os departamentos de uma direção?
- A que direção pertence um empregado?
- A que direção pertence um departamento?

Mas,

- A que departamento pertence um empregado?
- Quais os empregados de um departamento?

São questões que não podem ser respondidas pelo diagrama.

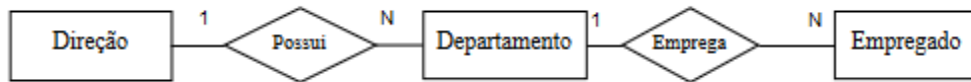
Porquê?



(o empregado e_1 pertence ao departamento dep_1 ou dep_2 ?)

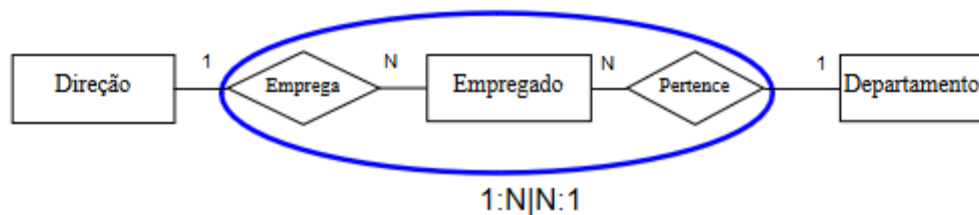
Uma possibilidade de remover o *fan trap* e responder às questões, é corrigir as associações e reestruturar o DEA original:

Tentativa 1:



Problema: Mas que fazer se um empregado pertencer a uma direção sem pertencer a nenhum dos seus departamentos?

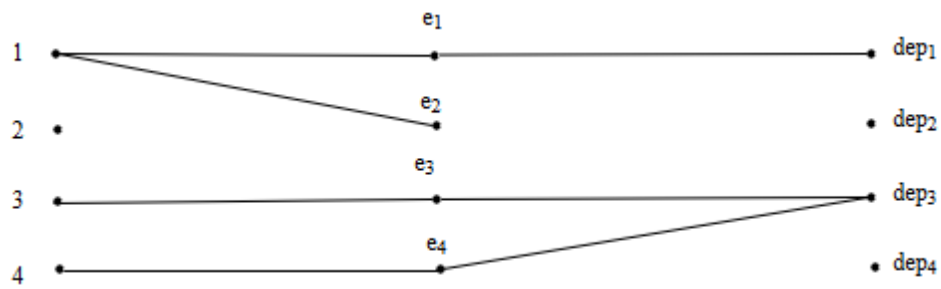
Tentativa 2:



Problema: A ligação de departamento a direção só se consegue se nesse departamento existir pelo menos um empregado.

Esta tentativa de solução depara-se com uma armadilha do tipo *chasm trap*.

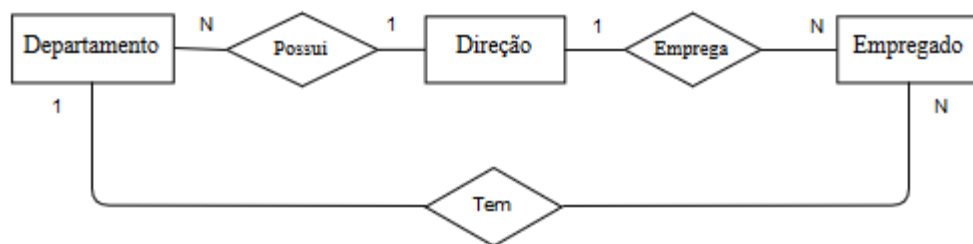
Chasm trap – Where a model suggests the existence of a relationship between entity types, but the pathway does not exist between certain entity occurrences.



(a que direção pertence o departamento dep_2 ?)

Tentativa 3 (solução):

Para responder a todas as questões, e também para resolver a *chasm trap*, é necessário considerar mais uma associação.



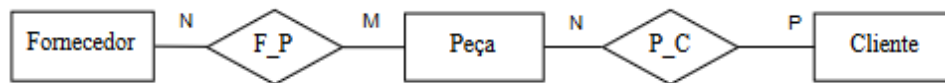
Exemplo 2:

Qualquer fornecedor fornece qualquer produto a qualquer cliente

Por exemplo:

- $F1$ fornece $P1$ para $C1$
- $F2$ fornece $P1$ para $C2$
- $F2$ fornece $P2$ para $C1$

Tentativa 1:



F1 fornece P1

P1 é fornecida a C1

F2 fornece P1

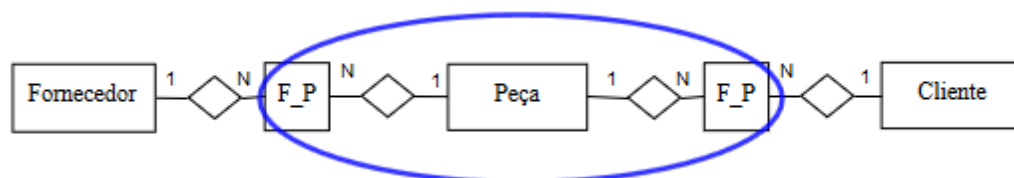
P1 é fornecida a C2

F2 fornece P2

P2 é fornecida a C1

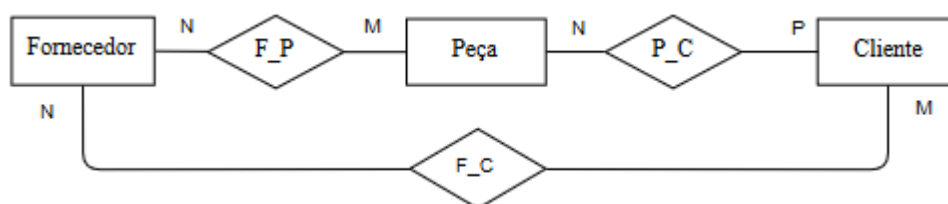
Problema: A pergunta “que fornecedores fornecem um dado cliente e o quê?” não pode ser respondida!

Decompondo vemos que temos um *fan trap*.



Tentativa 2:

Tentar uma abordagem semelhante ao exemplo 1: considerar uma ligação adicional entre *Fornecedor* e *Cliente*.



Exercício: esboce o diagrama de ocorrências correspondente

Podemos saber:

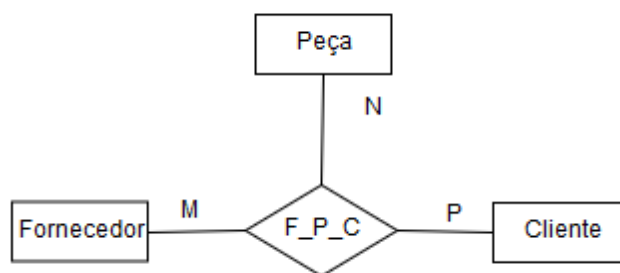
- Que fornecedores fornecem que peças?
- Que peças são fornecidas a que clientes?
- Que fornecedores fornecem que clientes?

Mas não, que fornecedores fornecem que peças a que clientes?

Não é possível ir de *Fornecedor* a *Cliente* passando por *Peça* sem “atravessar” uma estrutura $N:1|1:N$.

Exercício: decompondo as associações $M:N$.

Tentativa 3 (solução):

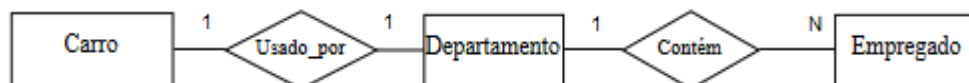


Como se viu anteriormente, as associações $M:N$ devem ser decompostas.

Exercícios:

1. Numa companhia cada departamento possui um carro que só pode ser usado por empregados do departamento devidamente autorizados.

Suponha a estrutura:



Os atributos de cada entidade são:

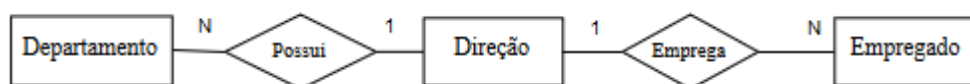
Carro: $N_registro$, *marca*

Departamento: $Nome_dep$, *localização*

Empregado: N_emp , *nome_emp*, *categoria*

- a) Se souber o número de um empregado autorizado, esta estrutura permitirá determinar qual o carro utilizado?
- b) Se souber o número de registo de um carro poderá saber que empregados estão autorizados a usá-lo?
- c) Como pode o diagrama ser estendido (por adição de uma associação) para que represente utilizadores autorizados?
- d) Como podem ser representados utilizadores autorizados se for permitido um novo atributo?
- e) Discuta as vantagens de substituir a associação “usado por” entre *Carro* e *Departamento* pela associação “Usado por” entre *Carro* e *Empregado*.
- f) Sugira uma alteração nas regras da empresa que mantendo o diagrama inalterado elimine a potencial ambiguidade entre *Carro* e *Empregado*.

2. Considere o DEA:



- Proponha duas restrições que tornem a associação entre *Departamento* e *Empregado* não ambígua.

3. A tabela abaixo mostra que professores lecionam que disciplinas a que estudantes:

Nome_prof	Nome_disc	N_estudante
João	Física	E1
João	Física	E2

Bruno	Física	E1
Bruno	Física	E2
Bruno	Biologia	E2

-

Assuma uma estrutura E-A com 3 associações binárias, *Professor_Disciplina*, *Disciplina_Estudante*, *Professor_Estudante*.

- Desenhe um diagrama de ocorrências e verifique que situação ambígua pode ocorrer.
- Desenhe um outro diagrama E-A

4. Cada disciplina é ensinada por um só professor, mas um professor pode dar várias disciplinas. Uma disciplina tem vários alunos e cada aluno pode ter várias disciplinas.

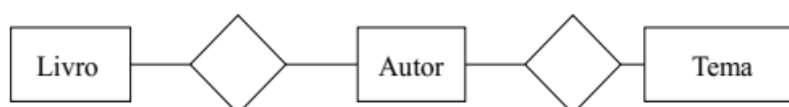
Proponha um diagrama E-A que contenha apenas duas associações.

5. Um modelo conceptual representa autores e a classificação por temas dos seus livros.

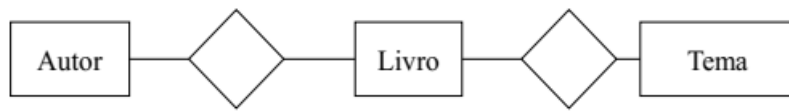
Não há dependências funcionais entre as chaves das entidades, *Autor*, *Livro*, *Tema*, mas um autor está sempre associado com todas as classificações aplicadas ao livro que escreveu.

Discuta as vantagens das estruturas seguintes. Qual escolheria?

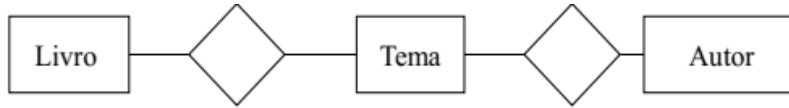
a)



b)



c)



4.7 Esquema Relacional

(Modelo Lógico!!!)

A modelação com recurso ao modelo entidade-associação, embora possa ser aplicada a outros modelos de dados, encontra no modelo relacional um terreno bastante fértil. Na verdade, a simplicidade e capacidade de representação do modelo entidade/associação contribuiu de forma significativa para o sucesso do modelo relacional.

Depois de obtido o DEA há que estabelecer o esquema relacional correspondente.

4.7.1 Associações 1:1

Suponhamos que os carros de uma companhia são atribuídos numa relação de 1 para 1:

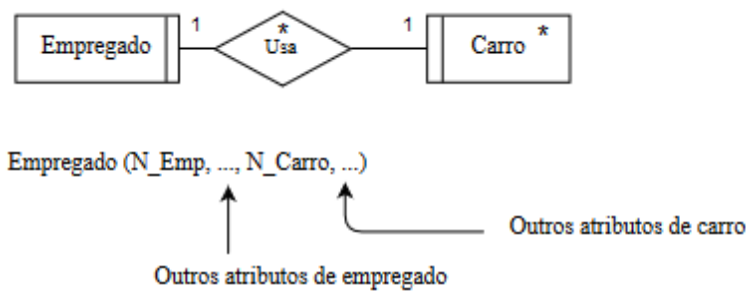
- Nenhum carro é partilhado entre empregados
- Nenhum empregado utiliza mais do que um carro



Caso 1: As duas entidades são obrigatórias

- Todo o carro é usado por um empregado
- Todo o empregado tem um carro da companhia

Basta uma tabela para representar a situação. Os atributos de carro podem ser vistos como atributos adicionais de empregado.



Exercício: Indique as chaves candidatas para a tabela empregado.

Caso 2: Só uma das entidades é obrigatória

- Todo o carro é usado por um empregado
- Nem todo o empregado tem um carro da companhia



Empregado (*N_Emp*, ...)

Carro (*N_Carro*, ..., *N_Emp*)

Os atributos de carro só são atributos de alguns empregados.

Duas tabelas, uma para cada entidade.

Colocar o identificador da entidade não obrigatória na tabela correspondente à entidade obrigatória.

Exercício: Porque não colocar os atributos de carro na tabela empregado?

Caso 3: Nenhuma das entidades é obrigatória

- Um empregado não tem necessariamente um carro;
- Um carro não tem necessariamente de ser usado.



Três tabelas: uma para cada entidade e uma para a associação.

Empregado (*N_Emp*, ...)

Carro (*N_Carro*, ...)

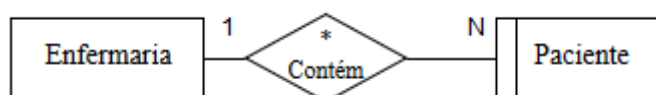
Usa (*N_Emp*, *N_Carro*)

4.7.2 Associações 1:N

Seja a afetação de doentes às enfermarias de um hospital (*só com registos de doentes atuais*):



Caso 1: Entidade do “lado N” obrigatória



Enfermaria (*Nome_Enf*, ...)

Paciente (*N_Paciente*, ..., *Nome_Enf*)

Caso 2: Entidade do “lado N” não-obrigatória

Suponhamos que alguns pacientes não pertencem a uma enfermaria:

Três tabelas:

Enfermaria (*Nome_Enf*, ...)

Paciente (*N_Paciente*, ..., *Nome_Enf*)

Contém (*N_Paciente*, *Nome_Enf*)

4.7.3 Associações M:N



Professor (Nome_Prof, ...)

Estudante (N_Est, ...)

Estuda_Com (Nome_Prof, N_Est)

Identificadores das associações:

- Geralmente o identificador da associação pode ser obtido por concatenação dos identificadores das entidades associadas;
- Uma exceção ocorre quando a mesma ocorrência de uma entidade é associada várias vezes com a mesma ocorrência de outra entidade.

Exercício:

Indique a chave da associação Consulta entre as entidades Médico e Paciente.

E se um paciente pudesse ter mais do que uma consulta no mesmo dia com o mesmo médico?

Regras gerais:

- Evitar ocorrências em que os identificadores de outras entidades tenham valores nulos.
- Não criar tabelas de modo a que identificadores de outras entidades se repitam.
- Criar tabelas para as associações apenas quando tal seja necessário para não violar os princípios anteriores.

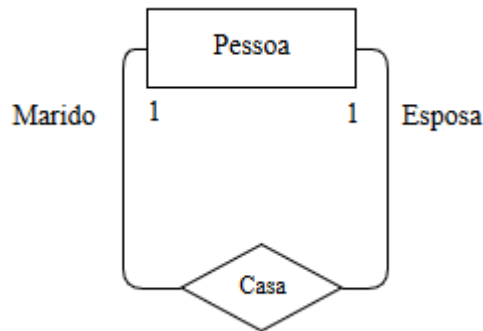
Uma associação sem atributos dá origem a uma tabela quando:

- É uma associação $1:N$ com entidade “do lado N ” não obrigatória;
- É uma associação $1:1$ com ambas as entidades não obrigatórias;
- É uma associação $M:N$.

4.8 Associações Unárias

4.8.1 Associações 1:1

Considere-se os casamentos numa sociedade onde uma do género masculino pode casar com uma pessoa do género feminino.



A entidade *Pessoa* tem “dois papéis”: *Marido* e *Esposa*.

Pessoa (*N_BI*, *Nome*, *Data_Nascimento*, ...)

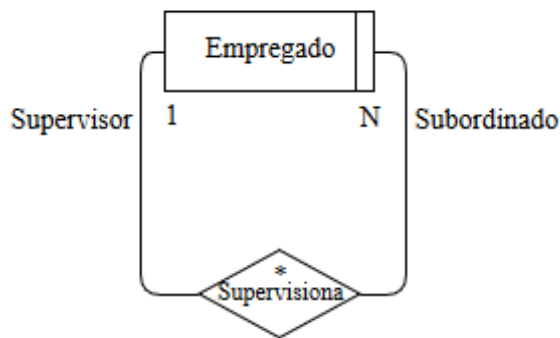
Casa (*BI_Esposa*, *BI_Marido*)

Exercício:

- Considere a possibilidade de dissolução do casamento (divórcio) e de novo casamento.*
- Considere o casamento entre pessoas do mesmo género.*

4.8.2 Associações 1:N

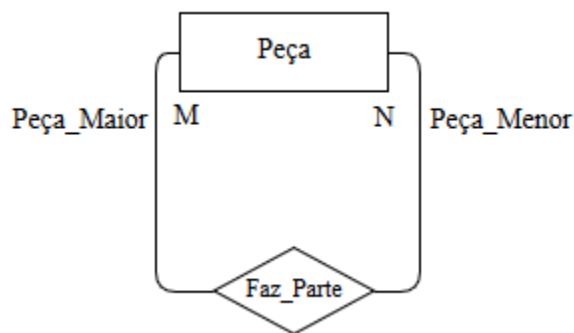
- Um empregado pode ser supervisor e/ ou subordinado;
- Alguns empregados não supervisionam outros, mas todo o empregado tem um supervisor (o empregado mais antigo supervisiona-se a si próprio).



Empregado (*N_Emp*, *Nome*, ..., *N_Emp_Chefe*)

4.8.3 Associações M:N

A fabricação de peças é feita a partir de outras peças.



Peça (*N_Peça*, *Designação*, ...)

Faz_Parte (*N_Peça_Maior*, *N_Peça_Menor*)

Exercícios:

1. Porque é que a entidade *Peça* não é obrigatória quer no papel de *Peça_maior* quer no papel de *Peça_Menor*?

2. Empregados de uma companhia podem opcionalmente ser membros de um clube desportivo da companhia.

Um empregado é identificado por um n° de empregado e um membro do clube tem um número de sócio.

- Desenhe um esquema de tabelas mostrando a associação entre as entidades *Empregado* e *Sócio*.
- Como é representada a associação?

3. Suponha que no exercício anterior os sócios do clube são identificados por *n_empregado*.

Qual seria a alteração à resposta anterior?

4. Suponha um modelo conceptual de uma sociedade monogâmica contendo as entidades, *Homem*, *Mulher* e a associação *Casamento*, mostrando os casamentos em vigor. Cada individuo é identificado pelo nº do bilhete de identidade.

- Qual o grau da associação *Casamento*?
- Qual o tipo de participação da cada entidade na associação?
- Construa o esquema de tabelas

5. Suponhamos que *Casamento* é tratado como uma entidade identificada por licença de casamento. Considerando só os casamentos atuais, qual é o grau da associação entre as entidades *Casamento* e *Pessoa* numa sociedade monogâmica?

6. Desenhe um esquema de tabelas para casamentos vigentes numa sociedade poliândrica. Use a entidade *Pessoa*, identificada por *BI*, e a associação *Casamento*.

7. Será a resposta à questão anterior alterada se, em vez da entidade *Pessoa*, se utilizar a entidade *Pessoa_casada*?

4.9 Afetação de atributos a “esboços” de esquemas de relação

Em princípio:

- Os atributos de uma entidade irão para a tabela correspondente;
- Os atributos de uma associação irão para a tabela correspondente quando exista.

Verificar se:

- Todos os atributos do sistema estão identificados;
- Existem atributos identificados sem que se saiba a que E/A pertencem.

Para cada tabela garantir que:

- Não contém grupos repetitivos;
- Quando a chave é composta qualquer dos restantes atributos deve depender da totalidade da chave;
- Não deve haver dependências entre atributos não chave.

Isto é, as tabelas devem estar normalizadas!

Pode haver a necessidade de reformular tabelas. Atenção a:

- Tabelas sem atributos;
- Não existência de qualquer tabela para conter um atributo.

Os ajustes no sistema de tabelas devem ser repercutidos no DEA!

Exercícios:

1. Seja o modelo:



Empregado (n_emp , $nome_emp$, $total_km_emp$)

Carro (n_carro , $marca$, $total_km_carro$, km_carro , n_emp)

Onde:

- *Km_carro* é o número de km que o atual utilizador já andou com o carro (no atual período de uso).
- *Total_km_emp* é o total de km que o empregado já andou em carros da companhia.

a) Suponha que a associação “Usa” é representada por uma tabela?

Reescreva o esquema de tabelas.

b) Suponha que a maior parte dos empregados nunca são autorizados a usar carro da companhia. Discuta cada um dos seguintes modelos:

A:

Empregado (n_emp , $nome_emp$, $total_km_emp$)

Carro (n_carro , $marca$, $total_km_carro$, km_carro , n_emp)

B:

Empregado (n_emp , $nome_emp$)

Carro (n_carro , $marca$, $total_km_carro$, km_carro ,
 $total_km_emp$, n_emp)

C:

Empregado (n_emp , $nome_emp$)

Emp_utiliza_carro (N_emp , $total_km_emp$)

Carro (n_carro , $marca$, $total_km_carro$, km_carro , n_emp)

2. Uma biblioteca guarda informação acerca dos seus livros e sócios e que livros estão emprestados a que sócios.

Cada exemplar é identificado por um número de exemplar e cada sócio por um número de sócio. Outros atributos são *Título*, *Data_de_aquisição*, *Preço_de_aquisição*, *Data_empréstimo*, *Nome_sócio*, *Limite_sócio*.

Um exemplar tem um só título. O *limite_sócio* é o número máximo de livros que um sócio pode ter emprestados ao mesmo tempo. Não é necessariamente igual para todos os sócios.

Construa o DEA, e respetivo o modelo relacional, usando *Sócio* e *Exemplar* como tipos de entidades.

4.10 Extensão do esquema relacional

Considere-se o modelo:



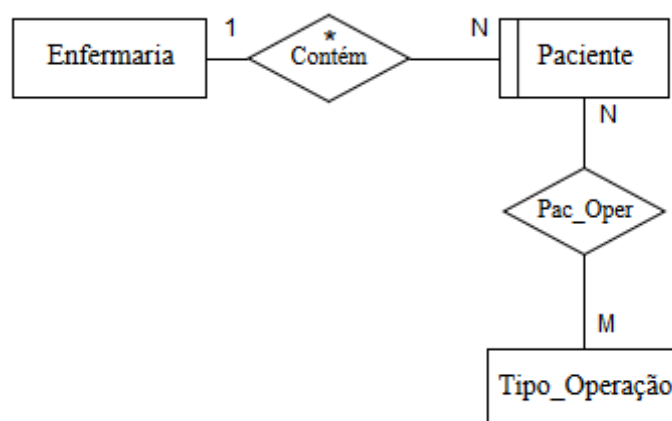
Enfermaria (Nome_Enf, Tipo_Enf, N_de_Camas)

Paciente (N_Paciente, Nome, Data_Nascimento,
Data_Internamento, Nome_Enf)

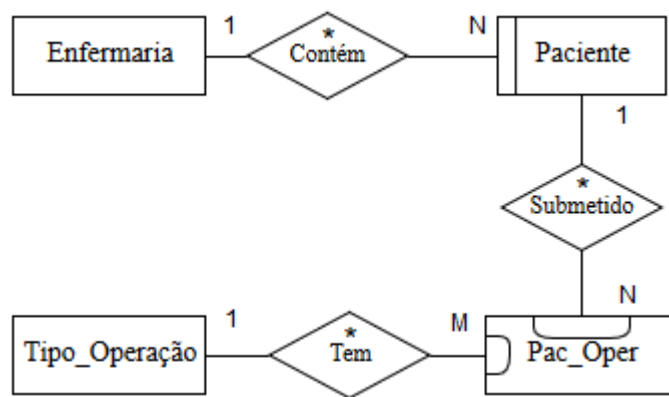
Suponha que é necessário incluir os atributos código de operação e nome de operação, onde código de operação determina nome de operação.

- Um paciente pode submeter-se a várias operações.

Redefina o modelo.



Decompondo a associação $M:N$:



Enfermaria (Nome_Enf, Tipo_Enf, N_de_Camas)

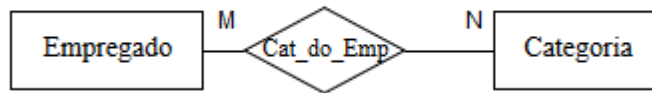
Paciente (N_Paciente, Nome, Data_Nascimento,
Data_Internamento, Nome_Enf)

Tipo_Operação (Cod_Operação, Designação)

Pac_Oper (N_Paciente, Cod_Operação)

4.11 Tabelas supérfluas

Exemplo:



Empregado (*N_Emp*, *Nome*, *endereço*)

Categoria (*Nome_Categoria*)

Cat_do_emp (*N_Emp*, *Nome_Categoria*)

A tabela da entidade *Categoria* é simplesmente uma lista de categorias.

Será necessário manter esta tabela?

Dois critérios:

1. No futuro será necessário introduzir no modelo atributos que dependam funcionalmente de *nome_categoria*?
(Por exemplo, *aumento_salário*)

Em caso afirmativo, *Categoria* deverá permanecer uma entidade.

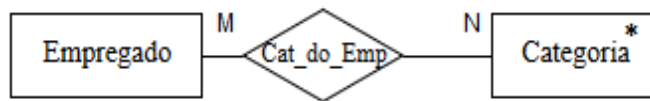
2. A entidade *Categoria* é obrigatória na associação?

Se a lista de categorias inclui valores que nenhum empregado possui, deverá manter-se.

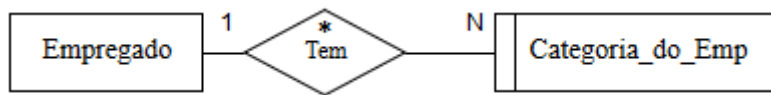
Caso contrário, se as únicas categorias que interessam são as dos empregados existentes pode eliminar-se a tabela.

Se não é necessário representar a entidade por uma tabela poder-se-á:

- a) Colocar um * na tabela correspondente:



b) Redesenhar o diagrama:



Cada ocorrência de *Categoria_do_Emp* é o nome de uma categoria de um determinado empregado.

4.12 Subentidades

Exemplo.

Suponha que cada empregado tem um *n_emp*, *nome_emp*, *endereço_emp*, *categoria*, *data_nascimento* e *salário*. Mas só os vendedores têm associada uma *quota_de_vendas* e um *bónus_de_vendas*.

Um vendedor que exceda a sua quota recebe um bónus além do seu salário. Quota e bónus são diferentes para cada empregado.

Modelos possíveis:

Vendedor (*n_emp*, *nome_emp*, *endereço_emp*, *categoria*,
data_nascimento, *salário*, *quota_vendas*, *bónus_vendas*)
Não_vendedor (*n_emp*, *nome_emp*, *endereço_emp*, *categoria*,
data_nascimento, *salário*)

Ou

Empregado (*n_emp*, *nome_emp*, *endereço_emp*, *categoria*,
data_nascimento, *salário*)
Empregado_Vendedor (*n_emp*, *quota_vendas*, *bónus_vendas*)
(associação 1:1)

Empregado_Vendedor pode ser visto como uma subentidade de *Empregado* porque a informação de *Empregado_Vendedor* juntamente com a de *Empregado* dá-nos toda a informação do vendedor.

No exemplo da secção anterior, *Categoria_do_Emp* pode ser vista como subentidade de empregado.

Exercícios:

1. Quais das seguintes restrições podem ser aplicadas ao uso do termo subentidade?
 - a. Uma subentidade tem de ter o mesmo identificador que a entidade principal.
 - b. O identificador da entidade principal tem de formar todo ou parte do identificador da subentidade.
 - c. O identificador de uma subentidade tem de ser parte do identificador da entidade principal.
 - d. A subentidade tem de ter participação obrigatória na associação com a entidade principal.
 - e. A subentidade tem de ter participação não obrigatória na associação com a entidade principal.

2. Dados acerca de empregados incluem, *n_emp*, *nome*, *endereço*, *data_nascimento*, *data_inicio_categoria*, *categoria*, *habilitação*, *salário_anual*, *pagamento_mensal*.
 - a. É necessário um histórico de categorias e data de início em cada uma delas.
 - b. Um empregado tem um só salário anual, mas pode ter até 12 valores de pagamento mensal representando o total pago nos 12 meses anteriores após deduções.
 - c. Um empregado pode possuir várias habilitações.

Desenhe um DEA usando uma entidade Empregado e várias subentidades para potenciais grupos de repetição.

4.13 Conceção final do esquema relacional

(1º nível de desenho)

Até agora fizemos a análise dos dados, isto é, estudamos as propriedades dos dados independentemente das transações que vão ser operadas nesses dados.

Falta analisar as transações que o modelo vai ter de suportar.

Passos para um desenho de 1º nível:

1. Esboçar um 1º esquema para ganhar sensibilidade ao problema.
2. Elaborar uma lista das transações que o modelo terá de suportar.
3. Elaborar uma lista de atributos.
4. Elaborar uma lista preliminar dos tipos de entidades que se possam seguramente identificar, e seleccionar para cada uma o identificador (isto é, a chave primária).
5. Desenhar um diagrama E-A mostrando as associações entre as entidades.

Incluir o grau das associações e os tipos de participação.

6. Fazer uma primeira verificação, ver se o diagrama suporta as transações e alterar o diagrama se necessário.

Olhar para possíveis situações de erro de interpretação e decidir se podem ser ignoradas.

7. Construir o esquema relacional correspondente ao diagrama E/A.
Eliminar todos os atributos usados no esquema, da lista de atributos.

8. Adicionar os atributos que restam às tabelas, da tal forma que estas permaneçam normalizadas.

Se a afetação de um atributo a uma tabela criar uma dependência funcional e esse atributo não for chave candidata, retirar o atributo determinante e os atributos determinados.

(colocar os atributos retirados, novamente na lista de atributos)

9. Se algum atributo não pode ser afetado às tabelas existentes, fazer a extensão do modelo. Se necessário voltar ao passo 5.

10. Decidir se alguns outros atributos ou transações deverão ser incluídos, nomeadamente para permitir futuros desenvolvimentos.

Para as transações voltar ao passo 6.

Para os atributos voltar ao passo 8.

11. Verificar se a escolha de entidades, associações e atributos permanece apropriada.

Verificar se as tabelas estão normalizadas.

Verificar se todas as transações são suportadas.

Se forem precisas alterações, repetir o procedimento, se necessário, desde o passo 1.

12. Apagar entidades supérfluas.

4.14 Exemplo – Biblioteca

Uma biblioteca guarda registo sobre os livros existentes e sobre os empréstimos aos seus sócios.

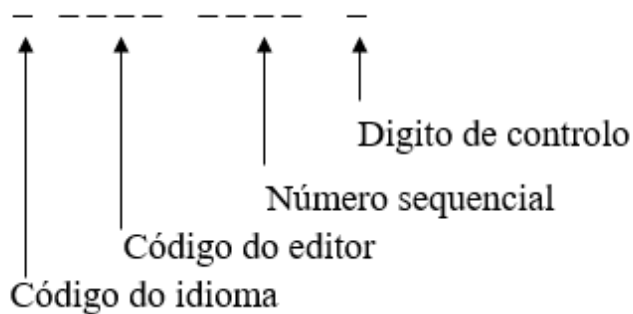
Cada sócio é identificado por um *número de sócio* e cada exemplar de livro por *número de exemplar* (pode existir mais do que um exemplar do mesmo livro).

O *nome* e *morada* de cada sócio é registado para tornar possível estabelecer comunicação, como, por exemplo, enviar avisos de devolução quando um empréstimo dura mais do que o período estabelecido.

As informações registadas sobre os livros são o *título*, *autores*, *editor*, *data publicação*, um código internacional – *ISBN*-, o *preço de compra* e o *preço corrente*.

O preço de compra é o preço que a biblioteca pagou pelo livro, enquanto que o preço corrente é o preço atual do livro no mercado.

O ISBN de um livro é um código de 10 dígitos com a seguinte estrutura:



Cada sócio pode ter em seu poder, em cada momento, um certo número de livros emprestados. Esse número é atualmente estabelecido em dois escalões em função da categoria de sócio (pleno ou correspondente).

Quando um sócio requisita, para empréstimo, um livro do qual não existe de momento nenhuma cópia disponível, é feita uma reserva que será

satisfeita quando possível. Reservas para o mesmo livro são satisfeitas por ordem de chegada.

Projeto

Passo 1 — Desenhe os seus esboços

Passo 2 — Uma 1ª lista de transações

1. Inserir detalhes de novos sócios
2. Inserir detalhes de novas aquisições
3. Fazer um empréstimo
4. Registrar a devolução de um empréstimo
5. Eliminar um sócio
6. Eliminar uma aquisição
7. Reservar um livro
8. Eliminar reserva
9. Alterar preço corrente
10. Enviar aviso quando termina o prazo de empréstimo

Passo 3 — Uma 1ª lista de atributos

*N_sócio, N_exemplar, nome_sócio, endereço_sócio, título,
nome_autor, nome_editor, data_publicação, ISBN, preço_compra,
preço_corrente, limite_empréstimo, tipo_sócio, data_empréstimo,
data_reserva.*

Seja $ISBNX = \{\text{cod_editor}, \text{n_série}\}$

Passo 4 - Entidades e chaves:

Sócio (n_sócio, ...)

Exemplar (n_exemplar, ...)

Passo 5 – DEA:



- Um sócio pode ter vários exemplares, mas um exemplar não pode estar emprestado a mais do que um sócio.
- Um exemplar não tem que necessariamente estar emprestado, nem um sócio tem de ter livros em seu poder

Passo 6 – Verificar diagrama ...

- As transações 1, 2, 5 e 6 consistem em criar ou eliminar ocorrências de *Sócio* e *Exemplar*.
- Empréstimos (t3 e t4) podem ser processados pela associação *Empréstimo*.
- *preço_corrente* deverá ser atributo de *Exemplar* por isso t9 pode ser feita.
- A informação para avisos a sócios (t10) poderá ser encontrada usando *Exemplar*, *Sócio* e *Empréstimo*.
- O único problema diz respeito à reserva de livros (t7 e t8).

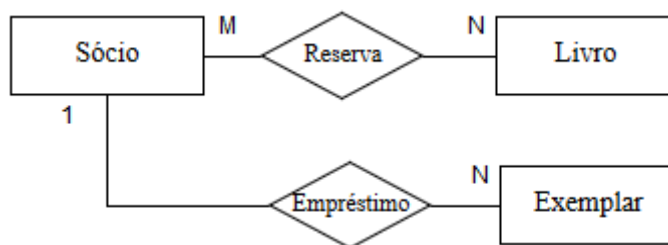
Um sócio pode fazer várias reservas e o mesmo livro pode ser reservado por vários sócios.

Uma **1ª hipótese** seria adicionar uma nova associação entre sócio e exemplar.

Mas, um sócio não reserva um determinado exemplar e sim um determinado livro.

Se há várias cópias de um livro não interessa qual o sócio vai receber.

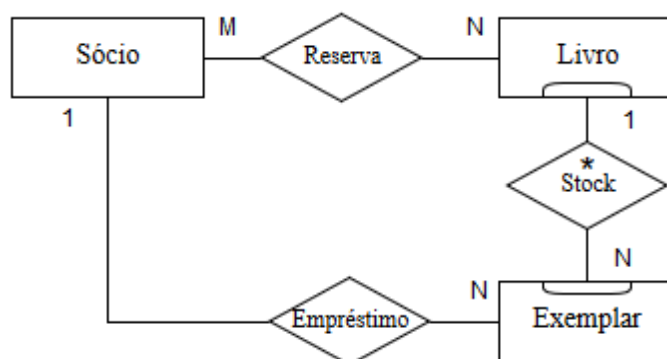
O diagrama pode ser alterado por introdução da entidade Livro e da associação reserva:



Mas, haverá uma situação de ambiguidade entre *Livro* e *Exemplar*:

- Quando um exemplar é devolvido é necessário saber se alguém reservou o livro.

Outra solução seria:



Livro (ISBNX, ...)

Passo 7 – Esquema relacional

Entidades:

Sócio (n_sócio, ...)

Exemplar (n_exemplar, ISBNX, ...)

Livro (ISBNX,)

Associações:

Empréstimo (n_exemplar, n_sócio, ...)

Reserva (n_socio, ISBNX, ...)

Passo 8 - Afetação de atributos

Sócio (n_sócio, nome_sócio, endereço_sócio)

Exemplar (n_exemplar, ISBNX, preço_compra)

Livro (ISBNX, título, data_publicação, preço_corrente)

Empréstimo (n_exemplar, n_socio, data_empréstimo)

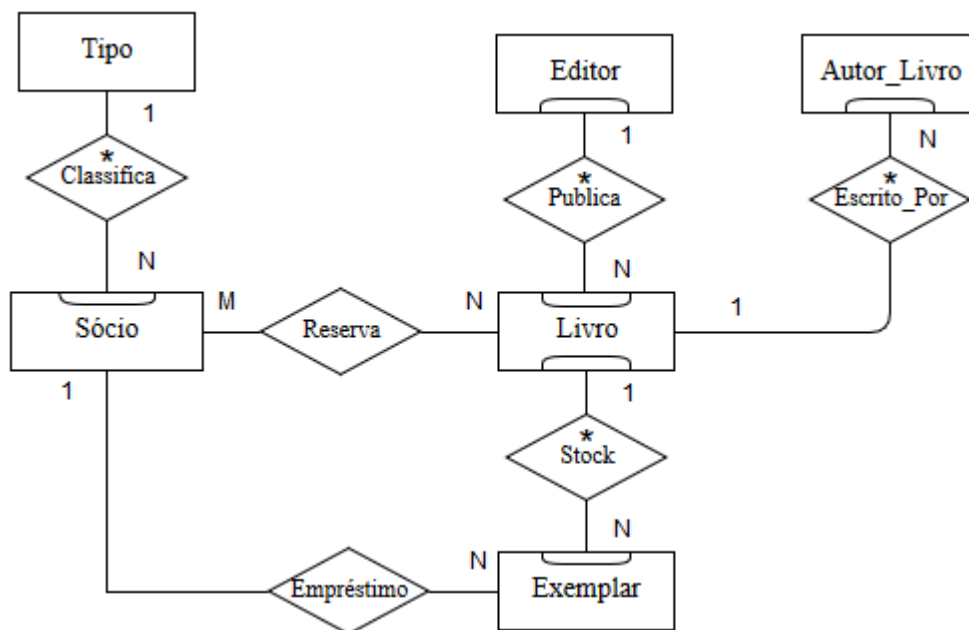
Reserva (n_sócio, ISBNX, data_reserva)

- Preço corrente não é atributo de exemplar (!).
- *Tipo_sócio* determina limite de empréstimo (!).
- O código do editor “contido” no ISBNX determina *nome_editor*.

Passo 9 – Afetar atributos que faltam ...

Falta afetar os atributos, *nome_autor*, *tipo_sócio*, *limite_empréstimo*, *nome_editor*.

- Se não interessam outros atributos de autor, para além do *Nome*, pode introduzir-se *Autor_Livro* como subentidade de *Livro*.
- É suposto que só são registados os editores de quem a biblioteca tem livros.



Sócio (*n_sócio*, *nome_sócio*, *endereço_sócio*, *tipo_sócio*)

Exemplar (*n_exemplar*, *ISBNX*, *preço_compra*)

Livro (*ISBNX*, *título*, *data_publicação*, *preço_corrente*, *cod_editor*)

Tipo (*tipo_sócio*, *limite_empréstimo*)

Editor (*cod_editor*, *nome_editor*)

Auto_Livro (*ISBNX*, *nome_autor*)

Empréstimo (*n_exemplar*, *n_sócio*, *data_empréstimo*)

Reserva (*n_sócio*, *ISBNX*, *data_reserva*)

Passo 10 - Outras transações que podem ser consideradas:

1. Inserir um novo livro
2. Eliminar um livro
3. Notificar o sócio de que a sua reserva já está em stock
4. Alterar o tipo de sócio
5. Quais os livros de um dado autor

O modelo permite responder.

Passo 11 – Verificar se as tabelas estão normalizadas, análise das transações...

- As tabelas estão normalizadas.

Deve ser feita uma análise detalhada de cada transação:

- Por exemplo, um empréstimo pode ser feito inserindo uma nova ocorrência na tabela *Empréstimo*,

ou

- Verificando primeiro se os números de sócio e de exemplar são válidos em *Sócio* e *Exemplar*.
- Eliminar o último exemplar de um livro implica não só eliminar a ocorrência do exemplar, mas também a ocorrência de *Livro* e talvez também as ocorrências de *Editor* e *Autor_Livro*, assim como verificar se há alguma reserva desse livro.

Grelha Transação/Atributo

Representar para cada transação a sequência de operações de Inserção (I), Atualização (A), Eliminação (E) e Consulta (C) a realizar.

E- Eliminar, C- Consultar, A – Atualizar, I – Inserir

E/A	Atributo	Empréstimo		Devolução		Alterar Tipo Sócio	
Sócio	n_sócio nome_sócio endereço_sócio tipo_sócio	C1 C		C4 C C		C1 C C	A2
Exemplar	N_exemplar ISBNX preço_compra	C2		C2 C			
Livro	ISBNX título data_publicação preço_corrente cod_editor			C5 C			
Tipo	tipo_sócio limite_empréstimo	C3 C					
Autor_Livro	ISBNX nome_autor			C6 C			
Editor	cod_editor nome_editor						
Empréstimo	n_exemplar n_sócio data_empréstimo	C4	I5 I5 I5	E1 E E			
Reserva	n_sócio ISBNX data_reserva			C C3 C3	E7 E7 E		

Notação:

- Uma transação pode aceder à mesma tabela em diferentes etapas, pelo que terá mais do que uma coluna.
- Um * indica que pode ser necessário aceder a várias linhas da tabela.
- Um índice mostra a ordem pela qual a tabela é acedida.
- Para consulta e eliminação o índice é aplicado ao(s) atributo(s) usado(s) para aceder à tabela.
- Para inserção e atualização o índice é aplicado a todos os atributos envolvidos.

Comentários:

Empréstimo

- Aceder a *Sócio* por *n_sócio* para validar *n_sócio* e saber *tipo_sócio*.
- Aceder a *Exemplar* por *n_exemplar* para validar *n_exemplar*.
- Aceder a *Tipo* por *tipo_sócio* para saber *limite_empréstimo*.
- Aceder a *Empréstimo* por *n_sócio*, contar nº de exemplares emprestados e verificar se é inferior ao *limite_empréstimo*. Em caso afirmativo inserir um novo empréstimo.

Devolução

- Eliminar ocorrência de empréstimo.
- Encontrar ISBNX para o número de exemplar devolvido.

- Encontrar o sócio com a mais antiga reserva do livro.
- Ler o nome e o endereço do sócio.
- Encontrar o título e autor do livro.
- Eliminar reserva.

Alterar Tipo de Sócio

- -Usar nº de sócio para obter nome de sócio e verificar o atual tipo de sócio.
- Se necessário alterar o tipo de sócio.

Passo 12

Não há tabelas supérfluas.

5 Normalização avançada

5.1 Dependências Multivalor (DM)

As dependências funcionais são um caso particular de um tipo mais geral de dependências lógicas, entre os atributos de uma relação, que são as dependências multivalor.

Exemplo:

Seja a relação *Inventário* (*item*, *departamento*, *cor*) e num dado instante a seguinte tabela:

Inventário

item	departamento	cor
c	1	castanho
t	1	preto
s	1	castanho
d	2	verde
s	2	castanho
b	2	vermelho
b	2	amarelo
b	2	azul
b	3	vermelho
b	3	amarelo
b	3	azul

- Um item pode ter mais do que uma cor e ser usado em mais do que um departamento.
- Um departamento pode usar mais do que um item em várias cores.
- Uma cor específica não determina o item nem o departamento.

(Só existem as dependências funcionais triviais)

Na tabela anterior existe uma dependência não funcional:

“Se um item existe numa dada cor e é usado nalgum departamento, então esse departamento usa todas as cores desse item”

Isto é,

- - O conjunto de cores de um item é o mesmo em qualquer departamento que usa esse item.
- O conjunto das cores de um item depende só do item e não do departamento onde é usado.
- O conjunto de departamentos que utiliza um item depende só do item e não das suas cores.

Dizemos que item **multidetermina** cor (item \twoheadrightarrow cor)

e que item multidetermina departamento (item \twoheadrightarrow departamento)

Seja a relação, $R(X, Y, Z)$ com $m + n + r$ atributos, onde

$$X = \{X_1, X_2, \dots, X_m\},$$

$$Y = \{Y_1, Y_2, \dots, Y_n\},$$

$$Z = \{Z_1, Z_2, \dots, Z_r\},$$

são conjuntos disjuntos de atributos.

Um m-tuplo $\{x_1, x_2, \dots, x_m\}$ de valores dos atributos X_1, X_2, \dots, X_m é denotado por x . Análogo para y e z .

Y_{xz} denota o conjunto de tuplos definido por

$$Y_{xz} = \{y : (x, y, z) \in R\}$$

Exemplo

$$cor_{c1} = \{castanho\}$$

$$cor_{t1} = \{preto\}$$

$$cor_{s1} = \{castanho\}$$

$$cor_{d2} = \{verde\}$$

$$cor_{s2} = \{castanho\}$$

$$cor_{b2} = \{vermelho, azul, amarelo\}$$

$$cor_{b3} = \{vermelho, azul, amarelo\}$$

5.1.1 Definição 1 (DM)

Numa relação $R(X, Y, Z)$, existe uma dependência Multivalor, $X \twoheadrightarrow Y$ sse $Y_{xz} = Y_{xz'}$ para quaisquer x, z e z' para os quais Y_{xz} e $Y_{xz'}$ são conjuntos não vazios de atributos.

Por exemplo, na relação *Inventário*

$$cor_{s1} = cor_{s2} = \{castanho\}$$

$$cor_{b2} = cor_{b3} = \{vermelho, azul, amarelo\}$$

$$item \twoheadrightarrow cor$$

$Y_{xz} = Y_{xz'}$, verifica-se sse sempre que (x, y, z) e (x, y', z') são tuplos de $R(X Y Z)$ também o são (x, y', z) e (x, y, z') .

5.1.2 Definição 2 (DM)

A dependência multivalor, $X \twoheadrightarrow Y$ existe em $R(X, Y Z)$ sse sempre que (x, y, z) e (x, y', z') são tuplos de $R(X Y Z)$ também o são (x, y', z) e (x, y, z') .

Por exemplo,

$(b, 2, \text{vermelho})$ e $(b, 3, \text{azul})$ são tuplos de *Inventário* assim como $(b, 2, \text{azul})$ e $(b, 3, \text{vermelho})$

X multidetermina Y se um valor de Y identifica um conjunto de valores de X independentemente de outros atributos da relação.

5.1.3 DFs e DM

Se X e Y são conjuntos disjuntos de atributos da relação $R(X Y Z)$ tal que exista $X \rightarrow Y$ então também $X \twoheadrightarrow Y$.

5.1.4 DM Complementares

Se a DM $X \twoheadrightarrow Y$ está definida na relação $R(X Y Z)$ então também existe $X \twoheadrightarrow Z$.

Na relação *Inventário*, se $Item \rightarrow Cor$ então $Item \rightarrow Departamento$ (e vice-versa)

De facto,

$$\begin{aligned} Departamento_{b, vermelho} &= Departamento_{b, amarelo} \\ &= Departamento_{b, azul} = \{ 2, 3 \} \end{aligned}$$

5.1.5 Separação de DM

Sejam

$$\pi_{\text{departamento, item}}(\text{Inventário}) \text{ e } \pi_{\text{item, cor}}(\text{Inventário})$$

Departamento	Item	Item	Cor
1	c	c	castanho
1	t	t	preto
1	s	s	castanho
2	d	d	verde
2	s	b	vermelho
2	b	b	azul
3	b	b	amarelo

Se fizermos a junção sobre item obtemos a relação inicial!

E se retirarmos o tuplo $(b, 2, vermelho)$ da tabela inicial?

5.1.6 Restabelecer uma relação pela junção das suas projeções.

Seja $R(X, Y, Z)$ uma relação, com X, Y e Z conjuntos disjuntos.

$R(X, Y, Z) = \pi_{X,Y}(R) \bowtie \pi_{X,Z}(R)$ sse existe a DM $X \twoheadrightarrow Y$.

Demonstração:

Já vimos que é sempre verdade que $R(X, Y, Z) \subseteq \pi_{X,Y}(R) \bowtie \pi_{X,Z}(R)$.

Seja $xyz \in R(X, Y, Z) \subseteq \pi_{X,Y}(R) \bowtie \pi_{X,Z}(R)$. Então existem y' e z' tais que xyz' e $xy'z \in R(X, Y, Z)$.

Mas $X \twoheadrightarrow Y$, donde, por definição de DM, xyz e $xy'z' \in R(X, Y, Z)$.

Portanto, $xyz \in R(X, Y, Z)$.

Provamos que se $X \twoheadrightarrow Y$ então $R(X, Y, Z) = \pi_{X,Y}(R) \bowtie \pi_{X,Z}(R)$.

Seja agora $R(X, Y, Z) = \pi_{X,Y}(R) \bowtie \pi_{X,Z}(R)$ e sejam xyz e $xy'z'$ quaisquer tuplos de $R(X, Y, Z)$.

Então xy e $xy' \in \pi_{X,Y}(R)$, com xz e $xz' \in \pi_{X,Z}(R)$.

Por junção de xy e xy' com xz e xz' obtemos xyz , xyz' , $xy'z$ e $xy'z'$.

Porque assumimos que $R(X, Y, Z) = \pi_{X,Y}(R) \bowtie \pi_{X,Z}(R)$

então $xy'z \in R(X, Y, Z)$ e $xyz' \in R(X, Y, Z)$.

Donde, pela definição de dependência multivalor, $X \twoheadrightarrow Y$ em $R(X, Y, Z)$.

5.1.7 DM triviais

As DM $X \twoheadrightarrow \phi$ e $X \twoheadrightarrow Y$ existem para qualquer relação $R(X, Y)$ onde Z e Y são conjuntos disjuntos de atributos.

5.2 Dependências de Junção (DJ)

5.2.1 Definição

Seja a relação $R(A_1, A_2, \dots, A_n)$ e $\{X_1, X_2, \dots, X_m\}$ uma coleção de subconjuntos de $\{A_1, A_2, \dots, A_n\}$ tal que $X_1 \cup X_2 \cup \dots \cup X_m = \{A_1, A_2, \dots, A_n\}$.

R satisfaz a dependência de junção (DJ) $*\{X_1, X_2, \dots, X_m\}$ sse

$$R = \pi_{X_1} \bowtie \pi_{X_2} \bowtie \dots \bowtie \pi_{X_m}.$$

Uma DJ é trivial se para algum $i=1, 2, \dots, m$, $X_i = \{A_1, A_2, \dots, A_n\}$.

Exemplo:

$R(\text{Fornecedor}, \text{Peça}, \text{Projeto})$

Fornecedor	Peça	Projeto
F1	P1	J2
F1	P2	J1
F2	P1	J1
F1	P1	J1

existe a dependência de junção,

$$*\{\{\text{Fornecedor}, \text{Peça}\}, \{\text{Peça}, \text{Projeto}\}, \{\text{Projeto}, \text{Fornecedor}\}\}$$

$\pi_{\text{Fornecedor, Peça}}(\mathbf{R})$	$\pi_{\text{Peça, Projeto}}(\mathbf{R})$	$\pi_{\text{Projeto, Fornecedor}}(\mathbf{R})$
Fornecedor	Peça	Projeto
F1	P1	J2
F1	P2	J1
F2	P1	J1

$\pi_{\text{Fornecedor, Peça}}(\mathbf{R}) \bowtie \pi_{\text{Peça, Projeto}}(\mathbf{R})$:

Fornecedor	Peça	Projeto
F1	P1	J2
F1	P1	J1
F1	P2	J1
F2	P1	J2
F2	P1	J1

$\pi_{\text{Fornecedor, Peça}}(\mathbf{R}) \bowtie \pi_{\text{Peça, Projeto}}(\mathbf{R}) \bowtie \pi_{\text{Projeto, Fornecedor}}(\mathbf{R})$:

Fornecedor	Peça	Projeto
F1	P1	J2
F1	P2	J1
F2	P1	J1
F1	P1	J1

5.2.2 Dependências de Junção e Dependências Multivalor

A relação $R(X, Y, Z)$ em que X, Y e Z são conjuntos não vazios e disjuntos, satisfaz a dependência de junção $\{XY, XZ\}$ sse

a DM $X \twoheadrightarrow Y$ for válida em R .

5.2.3 Dependências de Junção e Dependências baseadas em Chaves

Diz-se que a DJ $\ast \{X_1, X_2, \dots, X_m\}$ é o resultado de dependências baseadas em chaves de R sse cada Junção na expressão $\pi_{X_1} \bowtie \pi_{X_2} \bowtie \dots \bowtie \pi_{X_m}$ for efetuada sobre uma superchave, independentemente da ordem pela qual as junções são efetuadas.

Exemplo:

Seja, $R(A, B, C, D)$ em que $A \rightarrow BCD$ e $B \rightarrow ACD$

A DJ $\ast \{AB, AD, BC\}$ é baseada em chaves!

5.3 Quarta Forma Normal

Exemplo:

Seja a relação *Inventário* (Peça, Departamento, Cor)

Com $Peça \twoheadrightarrow Departamento$ e $Peça \twoheadrightarrow Cor$.

Está na FNBC (não existem dependências funcionais).

Mas, existem anomalias:

- Inserção: uma peça (com uma determinada cor) só pode ser inserida se existir um departamento que a use.
- Eliminação: eliminar um departamento significa eliminar todos os tuplos do departamento.
- Atualização: Se alterarmos a cor de uma peça tem de ser alterada a cor em todos os departamentos que usam essa peça.

Decompor em $E1 (\underline{Peça}, \underline{Departamento})$ e $E2 (\underline{Peça}, \underline{Cor})$.

A DM $Peça \rightarrow Departamento$ garante que $Inventário = E1 \bowtie E2$.

5.3.1 4ª Forma Normal (4FN)

Uma relação $R (X, Y, Z)$, com X, Y e Z conjuntos de atributos disjuntos, está na 4FN sse $\forall X \rightarrow Y$, não trivial: X é chave candidata de R .

5.3.2 4FN e FNBC

Se uma relação está na 4FN então necessariamente está na FNBC.

Demonstração:

Seja $R (A_1, A_2, \dots, A_n)$ na 4FN mas não na FNBC.

Então, existe uma DF não trivial $X \rightarrow Y$ e um atributo A tal que $X \not\rightarrow A$ (isto é, X não é chave candidata)

Seja YI o conjunto de atributos de Y que não pertencem a X .

$$X \rightarrow Y \Rightarrow X \rightarrow YI$$

$$X \rightarrow YI \wedge X \cap YI = \emptyset \Rightarrow X \rightarrow YI \text{ é não trivial,}$$

$YI \neq \emptyset$ e $X \cup YI$ não é o conjunto de todos os atributos,

$YI \neq \emptyset$ porque $YI = \emptyset$ significava que $X \rightarrow Y$ era trivial.

$X \cup Y1$ não é o conjunto de todos os atributos porque A não é de X nem de $Y1$.

(. se A pertencesse a X então X determinava A , o que contradiz a nossa hipótese;

. se A pertencesse a $Y1$, como $X \rightarrow Y$ e $Y1 \subseteq Y$ mais uma vez significava que X determinava A)

Portanto concluímos que R tem uma dependência multivalor não trivial $X \twoheadrightarrow Y1$ e um atributo A tal que $X \nrightarrow A$ (isto é, X não é chave candidata)

Logo R não está na 4FN.

Provámos que se R não está na FNBC então não está na 4FN, logo se R está na 4FN está necessariamente na FNBC.

Exemplo:

$R (\underline{Curso}, \underline{Professor}, \underline{Livro})$

$Curso \twoheadrightarrow Livro$

$Curso \twoheadrightarrow Professor$

Chave: $Curso, Professor, Livro$

Está em 3FN e em FNBC. Não está na 4^a FN

Decomposição:

$R1 (\underline{Curso}, \underline{Professor})$ e $R2 (\underline{Curso}, \underline{Livro})$

Estão na 4^a FN. $Curso$ não é chave candidata, mas a DM é trivial.

5.4 Quinta Forma Normal

(ou forma normal de projeção/junção – FNPJ)

5.4.1 Definição:

Uma relação R está na 5ªFN sse para toda a dependência de junção não trivial que se verifique em R , esta dependência é baseada em chaves.

Exemplo:

A relação R (*Agente*, *Companhia*, *Produto*), onde

- . um agente pode trabalhar para várias companhias e vender vários produtos,
- . uma companhia pode ter vários agentes e vende vários produtos,

obedece à dependência de junção

* $\{\{Agente, Companhia\}, \{Agente, Produto\}, \{Companhia, Produto\}\}$
que não é baseada em chaves.

A relação pode ser decomposta em

$\pi_{Agente, Companhia}(R)$

$\pi_{Agente, Produto}(R)$

$\pi_{Companhia, Produto}(R)$

5.4.2 Normalização em 5FN

Se uma relação R não está na 5FN então existe uma decomposição de R num conjunto de projeções que estão na 5FN e cuja junção natural restabelece a relação original.

Demonstração:

Suponhamos que R obedece à dependência de junção (DJ) $\{X_1, X_2, \dots, X_m\}$ que não é baseada em chaves de R .

No primeiro passo da normalização decompomos R no conjunto das suas projeções $\pi_{X_1}(R), \pi_{X_2}(R), \dots, \pi_{X_m}(R)$.

A junção destas projeções é igual a R porque assumimos a existência da DJ $\{X_1, X_2, \dots, X_m\}$.

Se a projeção $\pi_{X_i}(R)$ para algum $i = 1, 2, \dots, m$ não está na 5FN aplicamos o mesmo procedimento a $\pi_{X_i}(R)$.

O processo é finito porque no pior dos casos obtemos um conjunto de projeções binárias (obviamente na 5FN).

Referências

- Chen, P. (1976). The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1).
- Codd, E. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of ACM*, 13(6).
- Codd, E. (1974). Recent Investigations into Relational Data Base. *Proc. IFIP Congress*, North-Holland Pub Co., Amsterdam, pp. 1017-1021.
- Connolly, T., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. USA: Pearson.
- Coronel, C., & Morris, S. (2016). *Database Systems: Design, Implementation and Management*, 12 ed. USA: Cengage Learning.
- Date, C. (2004). *An Introduction to Database Systems*, 8th ed. USA: Pearson Education.
- Fagin, R. (1979). Multivalued Dependencies and a New Normal Form for Relational Databases. *ACM Transactions on Database System*, 13(3).
- Garcia-Molina, H., Ullman, J., & Widom, J. (2002). *Database Systems: The Complete Book*. Pearson, Prentice-Hall.
- Pereira, J. L. (1998). *Tecnologia de Bases de Dados*, 2ª Edição. FCA.
- Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems*, 3th Edition. McGraw-Hill.