

Universidade da Beira Interior

Departamento de Informática

Bases de Dados
2024/2025

J. Muranho, H. Proença, P. Prata, R. Cardoso

Notas de apoio às aulas teóricas

Versão 5.11, 2025-03-18

Histórico Alterações

5.3, 2021-09-28: Adicionado enunciado do trabalho prático do ano lectivo 2020/2021.

5.4, 2021-10-19: Consistência da base de dados (nova subsecção).

5.5, 2021-11-02: Reestruturação de capítulos (Cap 2 – Modelação de dados)

5.7, 2022-10-04: Adicionado enunciado do trabalho prático do ano lectivo 2021/2022.

Ficha da UC (BD - 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:

João Muranho

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 interage sobre a mesma.

Em suma, 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 SQL 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”, 14^a 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”, 14^a Edição, FCA, 2017, ISBN 978-972-722-829-4.
- 4) R. Ramakrishnan and J. Gehrke, Database Management Systems, McGraw-Hill, 2003.

Ficha UC (SGBD - 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:

João Muranho

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, aspectos 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

5.4 Orientado a Grafos

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 comprehende 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 interate sobre a mesma.

Em summa, 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.

Ficha UC (SIBD - EGI)

Unidade curricular /Curricular Unit:

Sistemas de Informação e Bases de Dados / Information Systems and 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:

João Muranho

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 dos sistemas de informação (SI) e das bases de dados (BD), incidindo na sua conceção e implementação. O seu objetivo principal é, portanto, preparar os alunos para entender, projetar e desenvolver sistemas de informação.

A nível das bases de dados, unidade curricular centra-se no modelo relacional, em especial sobre o modelo conceptual, a normalização, a linguagem SQL e a gestão da BD.

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

- Identificar as arquiteturas e topologias de SI;
- Entender a Gestão de SI (planeamento, desenvolvimento e exploração);
- Dada uma situação real, ou fictícia, desenvolver um modelo de dados (Diagrama Entidade-Associação e Esquema Relacional) que a represente;
- Normalizar e “desnormalizar” relações;
- Escolher um sistema de gestão de bases de dados em função do SI a desenvolver;
- Produzir o modelo físico da BD;
- Interrogar a base de dados (via SQL);
- Desenvolver aplicações sobre BD.

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

This course introduces the theme of the information systems (IS) and database (DB), focused on their design and implementation. Therefore, its main objective is to prepare students to understand, design and develop information systems.

At database level, the course focuses on the relational model, especially on the conceptual model, normalization, SQL and database management issues.

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

- Identify architectures and topologies of IS;
- Understand the IS Management (planning, development, and exploration);

- Given a real or hypothetical case, develop a suitable data model (Entity-Relationship Diagram and relational schema);
- Normalize and "de-normalize" relations;
- Choose a database management system that fulfills the needs of the IS to be developed;
- Produce the physical database model;
- Query the database (using SQL);
- Develop database applications.

Conteúdos programáticos:

1. Sistemas de Informação

1.1 Arquiteturas de Sistemas de Informação

1.2 Gestão de Sistemas de Informação

1.3 Desenvolvimento de Sistemas de Informação

2. 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)

2.2 Conceitos fundamentais

2.3 Modelos de dados (Hierárquico, Rede e Relacional. Estruturas de dados e linguagens de manipulação associadas)

3. Modelo Relacional

3.1 O modelo de dados

3.2 Álgebra relacional

3.3 Linguagens relacionais

3.4 Restrições de integridade

3.5 Transações

3.6 Dependências lógicas

4. Elaboração do modelo conceptual de uma base de dados

4.1 Modelo entidade-associação

4.2 Teoria da normalização

5. Desenvolvimento de aplicações sobre bases de dados

Syllabus:

1. Information systems

1.1 Architectures and Topologies of Information Systems

1.2 Information Systems Management

1.3 Information Systems Development

2. Introduction to database systems

2.1 Data files vs Desktop databases vs Client/server databases: advantages, disadvantages and when use (or not use)

2.2 Fundamental concepts

2.3 Data models (hierarchic; network; and relational. Data structures and manipulation languages)

- 3. The relational model.
 - 3.1 The data model
 - 3.2 Relational algebra
 - 3.3 Database query languages
 - 3.4 Integrity constraints
 - 3.5 Transactions
 - 3.6 Logical dependences
- 4. Conceptual database design
 - 4.1 Entity-Relationship modelling
 - 4.2 Normalization
- 5. Database applications development

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

Os conteúdos programáticos são desenvolvidos tendo em atenção o objetivo principal da unidade curricular: preparar os alunos para entender, projectar e desenvolver sistemas de informação. Assim, numa primeira fase são apresentados os conceitos gerais relacionados com a temática dos sistemas de informação, nomeadamente, o conceito de sistema de informação, o planeamento, desenvolvimento e exploração de sistemas de informação e também a integração de sistemas de informação. De seguida é focada a componente das bases de dados, sendo apresentadas algumas vantagens/desvantagens do desenvolvimento de aplicações baseadas em ficheiros (método de desenvolvimento já conhecido dos estudantes). São também apresentadas as soluções baseadas em bases de dados desktop e em bases de dados cliente servidor e é realçado o papel da nova entidade, o SGBD (Sistema de Gestão de Bases de Dados).

Depois de enquadrada a temática geral são apresentados os 3 modelos de dados (hierárquico, rede e relacional), o que permite aos estudantes começarem a assimilar conceitos e a falar o novo “dialecto”.

Assimilados os conceitos e conhecidas as linguagens relacionais, passa-se então para outro nível: a produção de modelos de dados dotados de boas propriedades. Nesta altura são trabalhadas as técnicas para a produção do modelo entidade-associação (DEA) e do respectivo esquema relacional. Em seguida são estudados as temáticas da normalização e do refinamento de modelos de dados existentes. O curso conclui-se com a implementação de algumas soluções informáticas para problemas hipotéticos.

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

The syllabus is focused on the course main objective: prepare students to understand, design and develop information systems. Thus, initially the general concepts are introduced, namely, the concept of information system, the management of information systems (planning, development and exploration), and also the integration of information systems. Then are focused the database system component, starting with the advantages/disadvantages of applications supported by traditional data files (method

already known to the students). Next, the desktop and client/server databases are introduced in the system developing equation and the role of the new entity, the DBMS (Database Management System) is highlighted.

Once the general theme is exposed, the focus is put on the data models (hierarchical, network and relational). The relational model is then studied in detail (adequate for a first database course).

Assimilated the relational concepts and known the relational languages, the attention is given to another subject: produce data models with good properties. At this point, the techniques to build entity-relationship (E-R) diagrams and its relational schema are studied. The normalization of relations is studied and applied to the refine existing data models. The course concludes with the implementation of some case studies.

Metodologias de ensino (avaliação incluída):

As aulas estão organizadas em aulas teóricas (T), para exposição dos conteúdos programáticos (diapositivos e escrita manual) e interacção dos 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. O trabalho prático (projecto) é desenvolvido em grupo.

A avaliação comprehende três componentes:

- Um teste escrito (8 valores);
- Preparação e apresentação de um tema relacionado com a disciplina (2 valores)
- Trabalho prático (10 valores).

Teaching methodologies (including assessment):

The course is structured with alternated theoretical (T) classes, for exposure of the syllabus 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.

Practical work (project) is developed in the group.

The evaluation consists of three components:

- One writing test (8 points);
- Preparation and presentation of a topic related to the course (2 points);
- Practical work (10 points).

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

A unidade curricular tem a duração de um semestre lectivo, envolvendo 64 horas de contacto, 92 horas de trabalho autónomo e 4 horas para avaliação (160 horas no total). As 32 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, donde, durante as aulas, podem complementar esse material com as explicações orais apresentadas.

As aulas práticas, em laboratório devidamente equipado, permitem aceder a bases de dados desktop e cliente/servidor. Os computadores estão apetrechados com o software necessário para o desenvolvimento de aplicações. Portanto, com as aulas práticas, os alunos resolvem exercícios sobre as diferentes temáticas, desenvolvem e exploram os diferentes tipos de bases de dados (MS ACCESS e MS SQL Server), formulam interrogações à base de dados e desenvolvem aplicações com acesso a bases de dados. Com a execução do trabalho prático, os alunos, para além do trabalho em equipa, têm a possibilidade de concretizar, faseadamente, todos os passos inerentes à concepção, análise e construção de uma base de dados e também ao desenvolvimento de uma aplicação que interactue sobre a mesma.

Em summa, a metodologia seguida é adequada e permite atingir os objectivos 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 64 hours of contact, 92 hours of autonomous work and 4 hours for evaluation (total: 160 hours).

The 32 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, taking place in a well-equipped laboratory, provides access to desktop and client/server 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 issues, develop and explore different kind of databases (MS ACCESS and MS SLQ Server), formulate database queries and develop database applications.

With the practical work, the students work as a team and have the opportunity to implement, in phases, the design and analysis of a database system and also 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

Bibliografia recomendada:

1. Thomas Connolly, Carolyn Begg and Anne Strachan, “Database Systems, A Practical Approach to Design, Implementation and Management”, 5th Edition, 2009. Addison-Wesley, ISBN: 978-0321523068.
2. David Avison and Guy Fitzgerald. Information Systems Development: Methodologies, Techniques and Tools., 2006, McGraw-Hill, ISBN: 978-0077114176.
- 3) Ralph M. Stair, George Reynolds. Fundamentals of Information Systems. Course Technology, 2014.

Outra bibliografia:

1. José Luís Pereira , Tecnologia de Bases de Dados, 3^a Edição, FCA; ISBN: 978-972-722-143-1
- 2) C. J. Date. “An Introduction to Database Systems”. Addison-Wesley, 8th Edition, 2003 (ISBN: 978-0321197849).
- 3) J. Hoffer, M. Prescott, F. McFadden. “Modern Database Management”. 11th Edition, 2012, Prentice Hall (ISBN: 978-0132662253).
- 4) Luís Damas, “SQL – Structured Query Language”, 6^a Edição, FCA, 2005 (ISBN 978-972-722-443-2).
- 5) Abraham Silberschatz, Henry Korth, S. Sudarshan. “Database System Concepts”, 6th Edition , 2010, McGraw-Hill (ISBN: 978-0-07-352332-3)
- 6) Ramez Elmasri, Shamkant B. Navathe. “Fundamentals of Database Systems”, 6th Edition. 2011, Addison-Wesley (ISBN: 978-0-136-08620-8).

Main bibliography

1. Thomas Connolly, Carolyn Begg and Anne Strachan, “Database Systems, A Practical Approach to Design, Implementation and Management”, 5th Edition, 2009. Addison-Wesley, ISBN: 978-0321523068.
 2. David Avison and Guy Fitzgerald. Information Systems Development: Methodologies, Techniques and Tools., 2006, McGraw-Hill, ISBN: 978-0077114176.
 - 3) Ralph M. Stair, George Reynolds. Fundamentals of Information Systems. Course Technology, 2014.
- Other:
1. José Luís Pereira , Tecnologia de Bases de Dados, 3^a Edição, FCA; ISBN: 978-972-722-143-1
 - 2) C. J. Date. “An Introduction to Database Systems”. Addison-Wesley, 8th Edition, 2003 (ISBN: 978-0321197849).
 - 3) J. Hoffer, M. Prescott, F. McFadden. “Modern Database Management”. 11th Edition, 2012, Prentice Hall (ISBN: 978-0132662253).
 - 4) Luís Damas, “SQL – Structured Query Language”, 6^a Edição, FCA, 2005 (ISBN 978-972-722-443-2).
 - 5) Abraham Silberschatz, Henry Korth, S. Sudarshan. “Database System Concepts”, 6th Edition , 2010, McGraw-Hill (ISBN: 978-0-07-352332-3)
 - 6) Ramez Elmasri, Shamkant B. Navathe. “Fundamentals of Database Systems”, 6th Edition. 2011, Addison-Wesley (ISBN: 978-0-136-08620-8).

Índice

1	Introdução.....	25
1.1	Sistemas de Armazenamento de Dados	25
1.1.1	Sistemas de Ficheiros	26
1.1.2	Sistemas de Bases de Dados.....	29
1.1.2.1	Níveis de abstração.....	30
1.1.2.2	Independência de dados	34
1.1.2.3	Arquitetura ANSI/SPARC	35
1.2	Objetivos e Capacidades de um SGBD	36
1.2.1	Linguagens da base de dados.....	37
1.2.2	Transações e propriedades ACID	38
1.2.3	Controlo de acesso.....	41
1.2.4	Tolerância a Falhas.....	42
1.3	Arquitetura cliente-servidor	43
1.4	Consistência da base de dados	44
1.5	Processo de desenvolvimento de sistemas de bases de dados	45
1.5.1	Etapas do desenvolvimento de sistemas de bases de dados	45
1.5.2	Atividades principais do desenvolvimento.....	47
1.5.3	Critérios de produção de um modelo de dados ótimo	48
2	Modelos de Dados	49
2.1	Introdução	49
2.1.1	Modelo Hierárquico.....	53
2.1.2	Modelo em Rede.....	56
2.1.3	Modelo Relacional.....	60
2.1.3.1	Álgebra Relacional	64
2.1.3.2	Chaves	69
2.2	Modelo Relacional	72
2.2.1	Estrutura de Dados Relacional	72
2.2.1.1	Modelo Conceptual de Dados	72
2.2.1.2	Entidades, Atributos e Domínios	73
2.2.1.3	Representação de entidades por tuplos.....	74
2.2.1.4	Relação	76
2.2.1.5	Base de dados relacional e esquema relacional.....	76
2.2.2	Álgebra Relacional	78
2.2.2.1	Projeção	79
2.2.2.2	Restrição (ou seleção)	81
2.2.2.3	Operações com conjuntos.....	82
2.2.2.4	Operações de junção.....	84
2.2.2.5	Divisão	90
2.2.2.6	Renomear (rebatizar).....	93
2.2.2.7	Operações de agregação e de agrupamento.....	93

2.2.2.8	Combinação de operações para formar <i>Queries</i>	97
2.3	SGBD relacional	99
2.3.1	Estrutura de um SGBD relacional	99
2.3.2	Processamento de consultas.....	101
2.3.3	Catálogo.....	102
2.3.4	Fases do processamento de consultas.....	104
2.3.5	Otimização de consultas	106
2.3.6	As doze regras de Codd	110
2.4	Linguagens Relacionais	112
2.4.1	História do SQL:.....	113
2.4.2	Query block	113
2.4.3	Projeção	114
2.4.4	Restrição	115
2.4.5	Junção (Equijunção)	117
2.4.6	Produto Cartesiano	118
2.4.7	União, Intersecção e Diferença.....	118
2.4.8	Divisão.....	119
2.4.9	Funções Standard.....	120
2.4.10	Actualizações	121
2.4.10.1	Inserção	121
2.4.10.2	Eliminação.....	122
2.4.10.3	Actualização	122
2.5	Restrições de integridade	123
2.5.1	Integridade de domínio	124
2.5.2	Integridade de entidade.....	124
2.5.3	Integridade referencial	125
2.5.4	Regras de negócio.....	126
3	Teoria da Normalização	127
3.1	Dados redundantes	127
3.2	Dependências Funcionais	129
3.2.1	Definição de Dependência Funcional (DF)	130
3.2.2	Diagrama de Dependência Funcional.....	130
3.2.3	Chave (candidata)	131
3.2.4	Superchave.....	131
3.2.5	Toda a relação tem uma chave	132
3.2.6	Chave Primária	132
3.2.7	Propriedades básicas das DFs.....	132
3.2.7.1	Unicidade	132
3.2.7.2	Reflexibilidade	132
3.2.7.3	Transitividade.....	132
3.2.7.4	Aumento	132
3.2.7.5	Axiomas de Armstrong	133
3.2.8	Propriedades derivadas das DFs	134
3.2.8.1	Distributividade (decomposição)	134
3.2.8.2	União	134
3.2.8.3	Pseudotransitividade.....	134
3.2.9	DF trivial	135

3.2.10	Fecho de um conjunto de DFs	135
3.2.11	Fecho de um conjunto de atributos	136
3.2.12	Cobertura e equivalência	137
3.2.13	Chaves e Fecho	139
3.2.14	Perda de informação	139
3.2.15	Decomposição sem perda (<i>Lossless Join</i>)	141
3.3	Normalização	143
3.3.1	Primeira Forma Normal (1FN)	144
3.3.2	Segunda Forma Normal (2FN)	146
3.3.2.1	DF Elementar	147
3.3.2.2	DF Parcial.....	148
3.3.2.3	Segunda Forma Normal (2FN).....	148
3.3.2.4	Casos especiais de relações na 2FN:	149
3.3.2.5	Decomposição em 2FN	150
3.3.3	Terceira Forma Normal (3FN)	151
3.3.3.1	DF Direta.....	153
3.3.3.2	Terceira Forma Normal (3FN)	153
3.3.3.3	3FN e 2FN	154
3.3.3.4	Decomposição em 3FN	154
3.3.4	Forma Normal de Boyce-Codd (FNBC)	156
3.3.4.1	Definição (FNBC)	157
3.3.4.2	FNBC e 3FN.....	161
3.3.4.3	Decomposição em FNBC	161
3.3.5	Preservação de dependências.....	162
4	Modelação de dados (DEA)	165
4.1	Introdução	165
4.2	Modelo Entidade-Associação	167
4.3	Propriedades das associações.....	170
4.3.1	Grau de uma associação.	170
4.3.1.1	Associação 1:1.....	170
4.3.1.2	Associação 1:N.....	171
4.3.1.3	Associação M:N.....	172
4.3.2	Tipo de participação	174
4.4	Decomposição de Associações M:N	177
4.5	Associações Complexas.....	181
4.5.1	Grau 1:1:1	182
4.5.2	Grau 1:1:N	183
4.6	Situações Ambíguas.....	184
4.7	Associações Unárias	193
4.7.1	Associações 1:1	193
4.7.2	Associações 1:N	193
4.7.3	Associações M:N	194
5	Modelação (modelo lógico)	195
5.1	Esquema Relacional.....	195

5.1.1	Associações 1:1	195
5.1.2	Associações 1:N	197
5.1.3	Associações M:N.....	198
5.2	Associações Unárias	199
5.2.1	Associações 1:1	199
5.2.2	Associações 1:N	200
5.2.3	Associações M:N.....	200
5.3	Afetação de atributos a “esboços” de esquemas de relação.....	202
5.4	Extensão do esquema relacional	205
5.5	Tabelas supérfluas.....	207
5.6	Subentidades	208
5.7	Conceção final do esquema relacional.....	210
5.8	Exemplo – Biblioteca.....	212
6	Normalização avançada.....	223
6.1	Dependências Multivalor (DM).....	223
6.1.1	Definição 1 (DM)	225
6.1.2	Definição 2 (DM)	226
6.1.3	DFs e DM	226
6.1.4	DM Complementares.....	226
6.1.5	Separação de DM.....	227
6.1.6	Restabelecer uma relação pela junção das suas projeções.	228
6.1.7	DM triviais.....	229
6.2	Dependências de Junção (DJ)	229
6.2.1	Definição	229
6.2.2	Dependências de Junção e Dependências Multivalor.....	230
6.2.3	Dependências de Junção e Dependências baseadas em Chaves.....	231
6.3	Quarta Forma Normal	231
6.3.1	4 ^a Forma Normal (4FN).....	232
6.3.2	4FN e FNBC.....	232
6.4	Quinta Forma Normal	234
6.4.1	Definição:	234
6.4.2	Normalização em 5FN	235
Referências	237	
Apêndice – Trabalhos Práticos (anos anteriores).....	239	
Ano lectivo 2013/2014	240	
Ano lectivo 2014/2015	247	
Ano lectivo 2015/2016	251	
Ano lectivo 2016/2017	257	
Ano lectivo 2017/2018	261	
Ano lectivo 2019/2020	265	

Ano lectivo 2020/2021	269
Ano lectivo 2021/2022	274
Ano lectivo 2022/2023 (EI).....	279
Ano lectivo 2023/2024 (EI).....	285

1 Introdução

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 accedemos 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 noutras 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 noutra local ou se este estiver armazenado noutra 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 e 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.

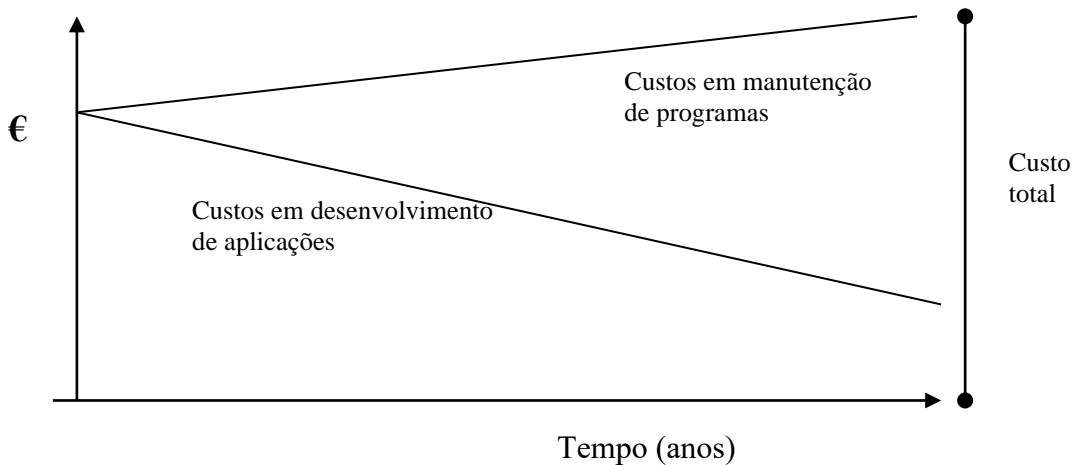
Uma vez que o relacionamento entre os dados é feito ao nível das aplicações, estes permanecem “isolados” em cada componente (ficheiro).

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

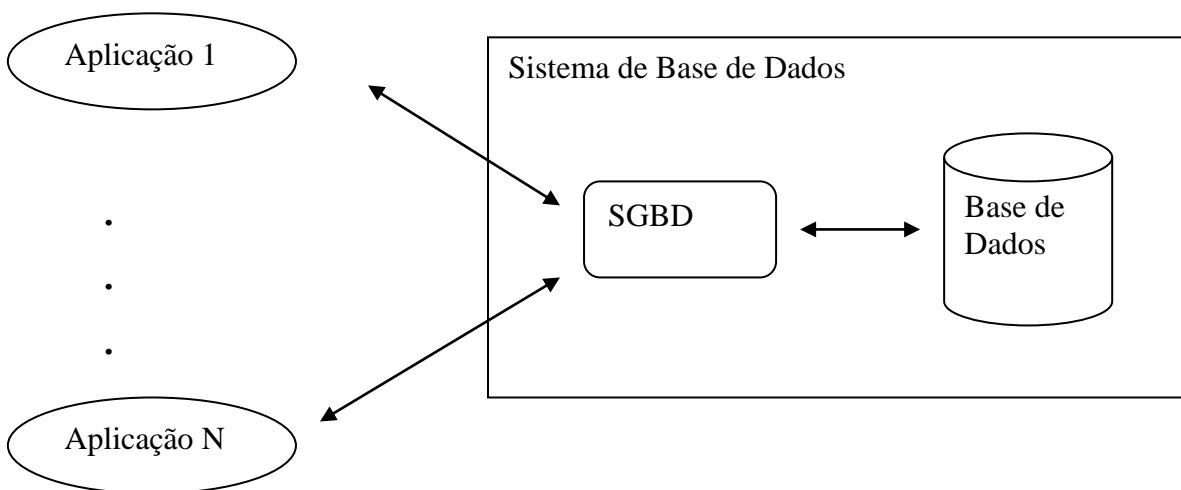
Um sistema de base de dados tenta baixar os custos de manutenção através da separação entre a forma como os dados são percebidos pelo programador e a forma como esses dados são armazenados fisicamente.

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 de 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 dos dados 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 SGBD e do sistema operativo no nível físico não estão claramente estabelecidas e variam de sistema para sistema. Alguns SGBD 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 ⇒ 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 aspectos 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”)

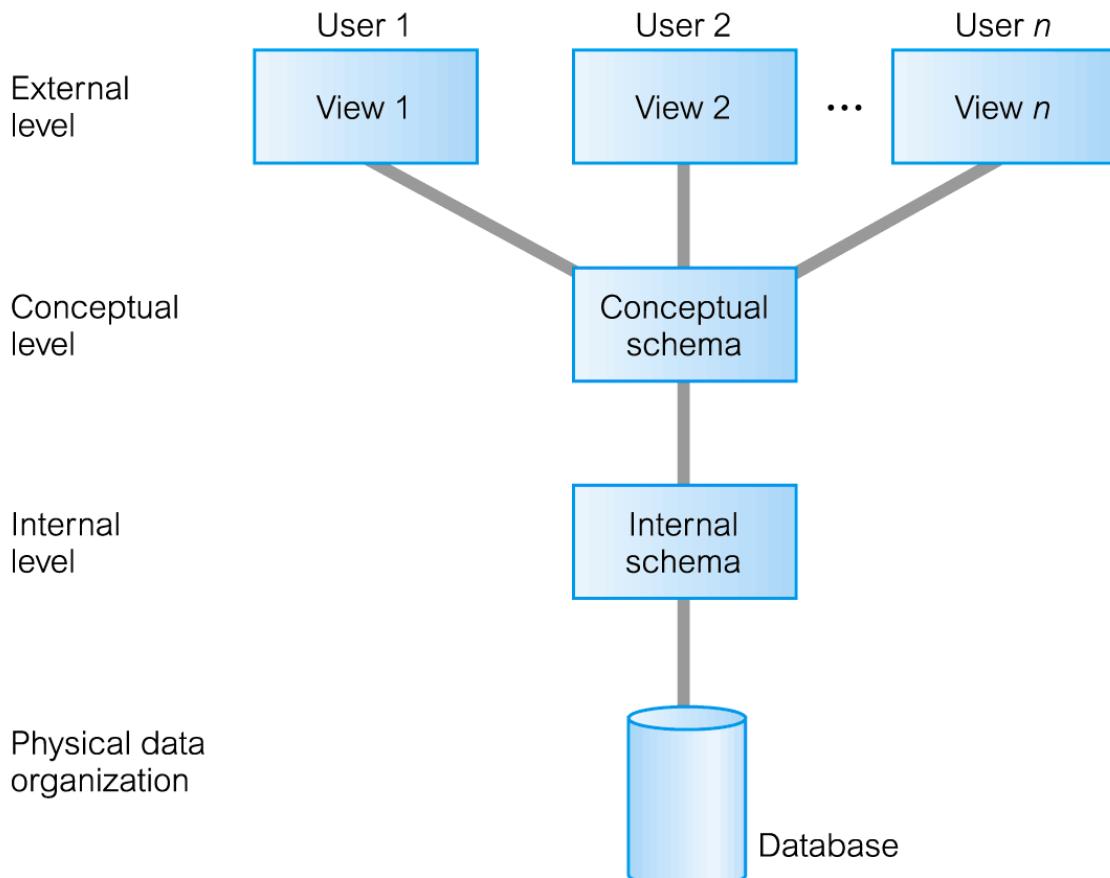
Eliminação de informação?

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?

Resposta:

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

¹ (Connolly & Begg, 2015), pág. 85.

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 e propriedades ACID

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 excepcional 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 operações 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.

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 excepcional diz-se que foi executado o *COMMIT* da transação.

Na ocorrência de alguma situação excepcional 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.

1.2.3 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.

1.2.4 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.

1.3 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.

Uma base de dados distribuída 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.

1.4 Consistência da base de dados

Algumas noções sobre consistência e integridade:

Inconsistência de dados: existe inconsistência de dados quando os mesmos dados estão armazenados em diferentes locais, mas com valores diferentes.

Integridade dos dados: a integridade dos dados define-se como o estado onde todos os dados na base de dados estão consistentes com o mundo real.

Restrições de integridade: são regras que definem que dados são válidos. As restrições de integridade asseguram que alterações feitas na base de dados por utilizadores autorizados não levam a perda da consistência dos dados. Portanto, as restrições de integridade protegem a base de dados de danos acidentais.

Estado consistente de uma BD: estado da base de dados onde todas as restrições de integridade são satisfeitas.

Questão:

Explique o que se entende por consistência de uma base de dados. A quem compete assegurar a consistência, ao utilizador ou ao SGBD?

R:

Consistência: estado da base de dados no qual todas as restrições de integridade são satisfeitas.

Para assegurar a consistência da base de dados, as transações devem iniciar-se com a base de dados num estado de consistência conhecido.

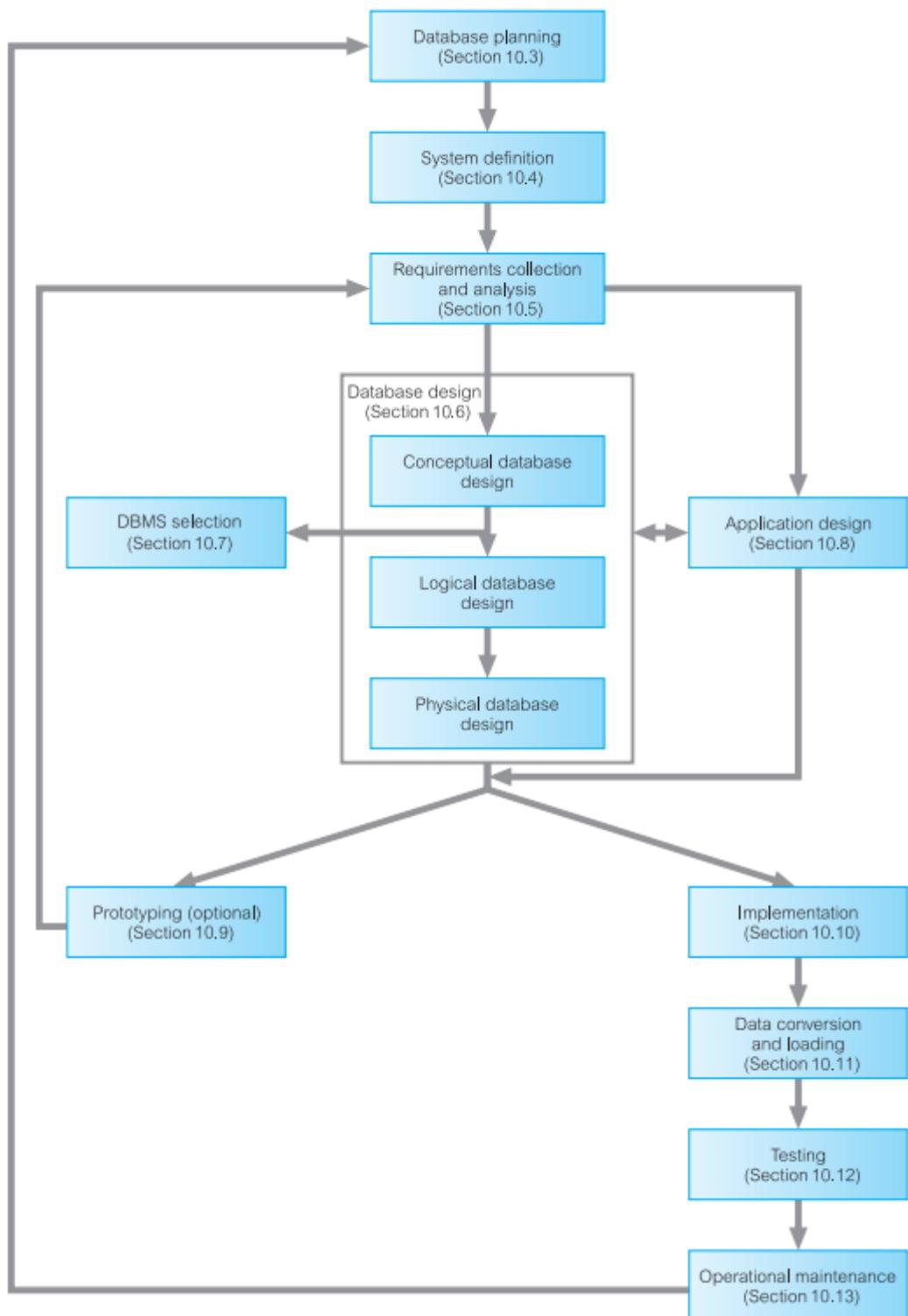
Uma transação faz evoluir a base de dados de um estado consistente para um novo estado de consistência.

Responsabilidade: ao programador, que codifica a transação, compete assegurar a consistência de cada transação isolada, pois é o programador que conhece as regras do mundo real.

Ao SGBD compete controlar a interação entre transações concorrentes para prevenir que destruam a consistência da base de dados. Nesta tarefa o SGBD usa uma variedade de mecanismos designados por esquemas de controlo de concorrência.

1.5 Processo de desenvolvimento de sistemas de bases de dados³

1.5.1 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.

1.5.2 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.

1.5.3 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>	Ability 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

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

Modelo Hierárquico

Modelo em Rede ou *Codasyl*

2^a Geração

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

3^o Geração

Modelo *Object - Oriented*

...

Suponhamos a **Base de Dados Exemplo:**

Obra

N_obra
Nome_obra
Data_de_inicio
Data_de_fim
Orçamento

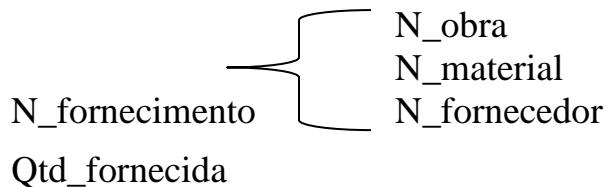
Fornecedor

N_fornecedor
Nome_fornecedor
Morada

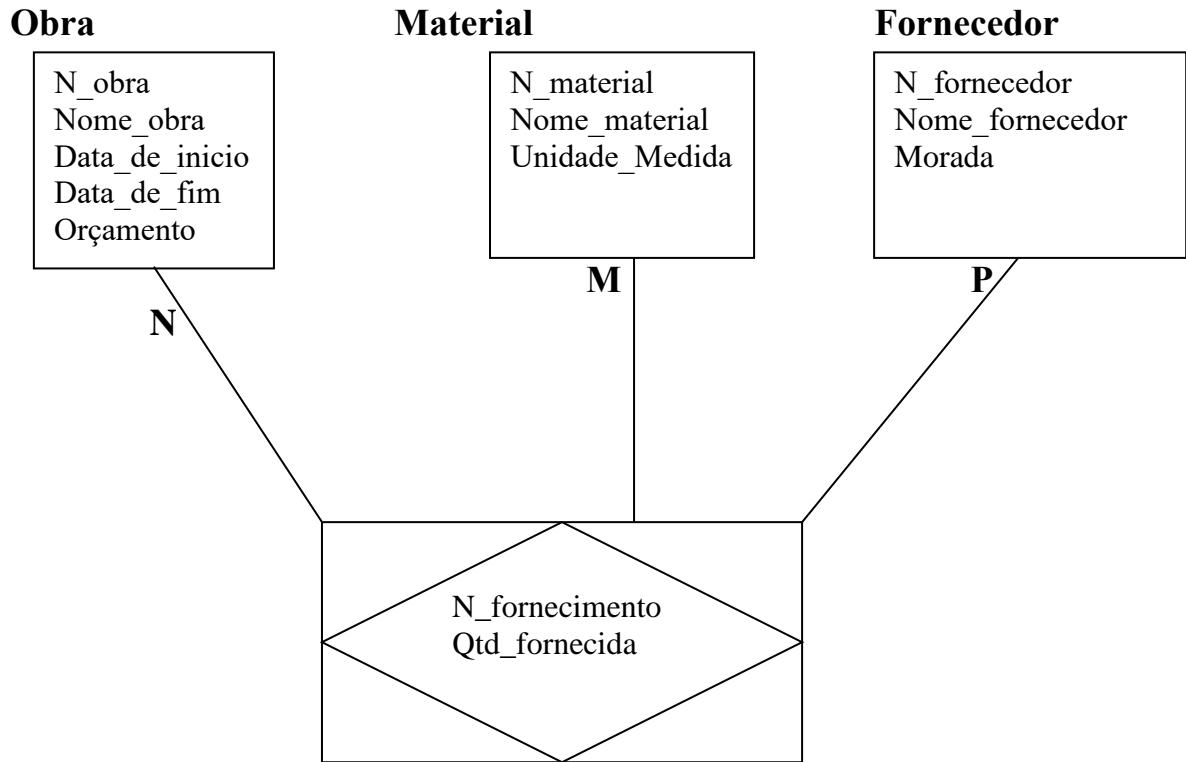
Material

N_material
Nome_material
Unidade_medida

Fornecimento



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 $M1$ 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 $F5$.

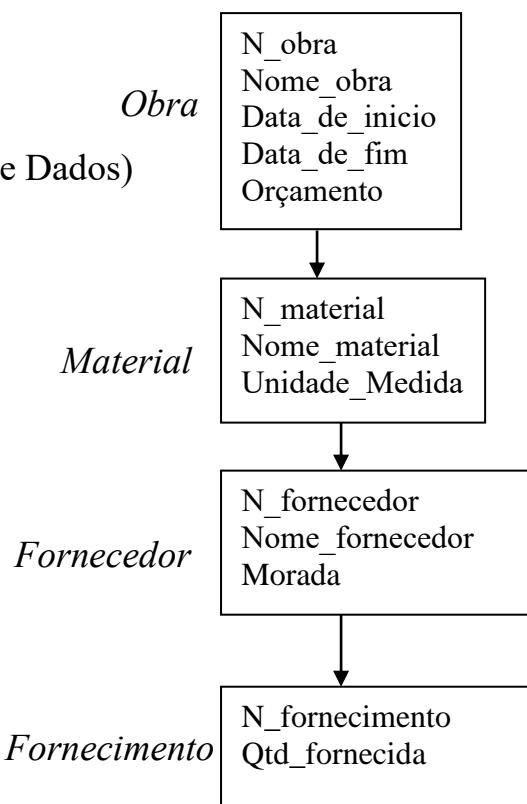
A3: Modificar a morada do fornecedor $F3$ para “Covilhã”

2.1.1 Modelo Hierárquico

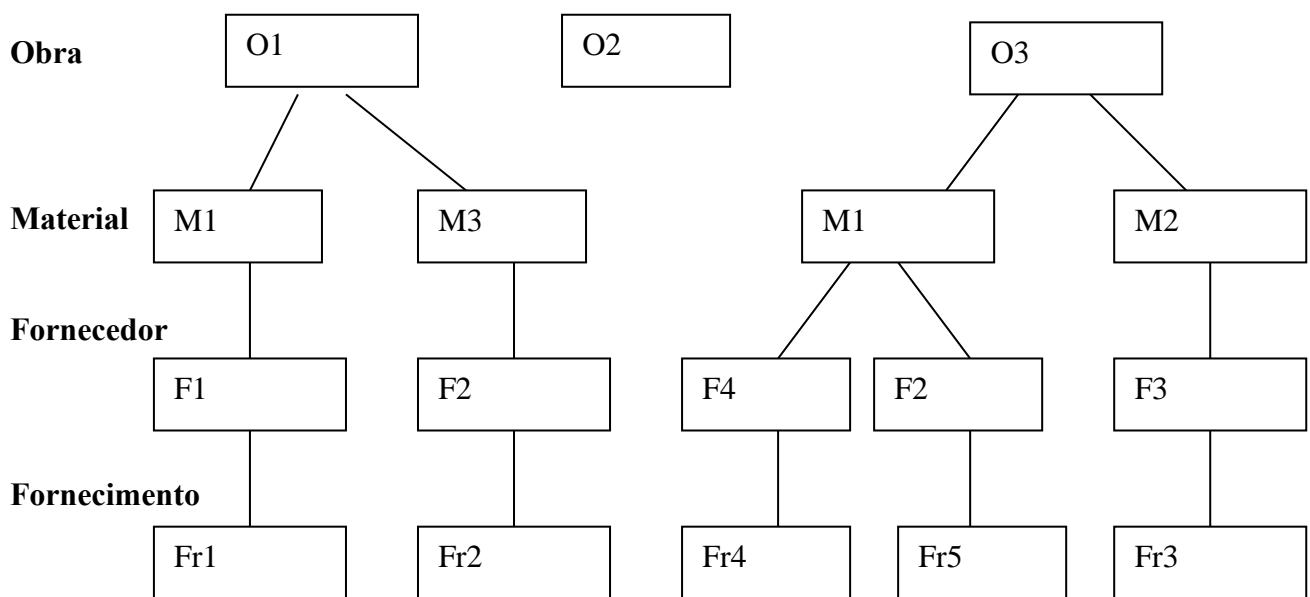
Um esquema possível:

Obra

(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 *M1* para a obra *O1*?

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”

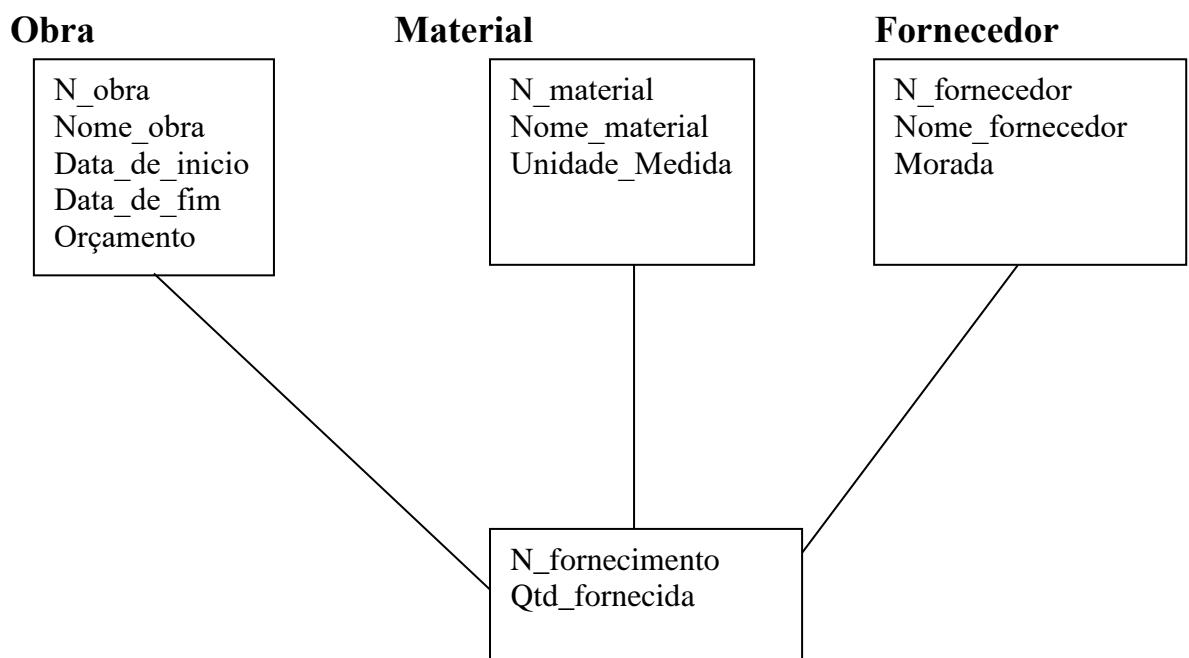
...

2.1.2 Modelo em Rede

Definido pelo DBTG (Data Base Task Group) da CODASYL (**COnference on Data Systems Languages**)

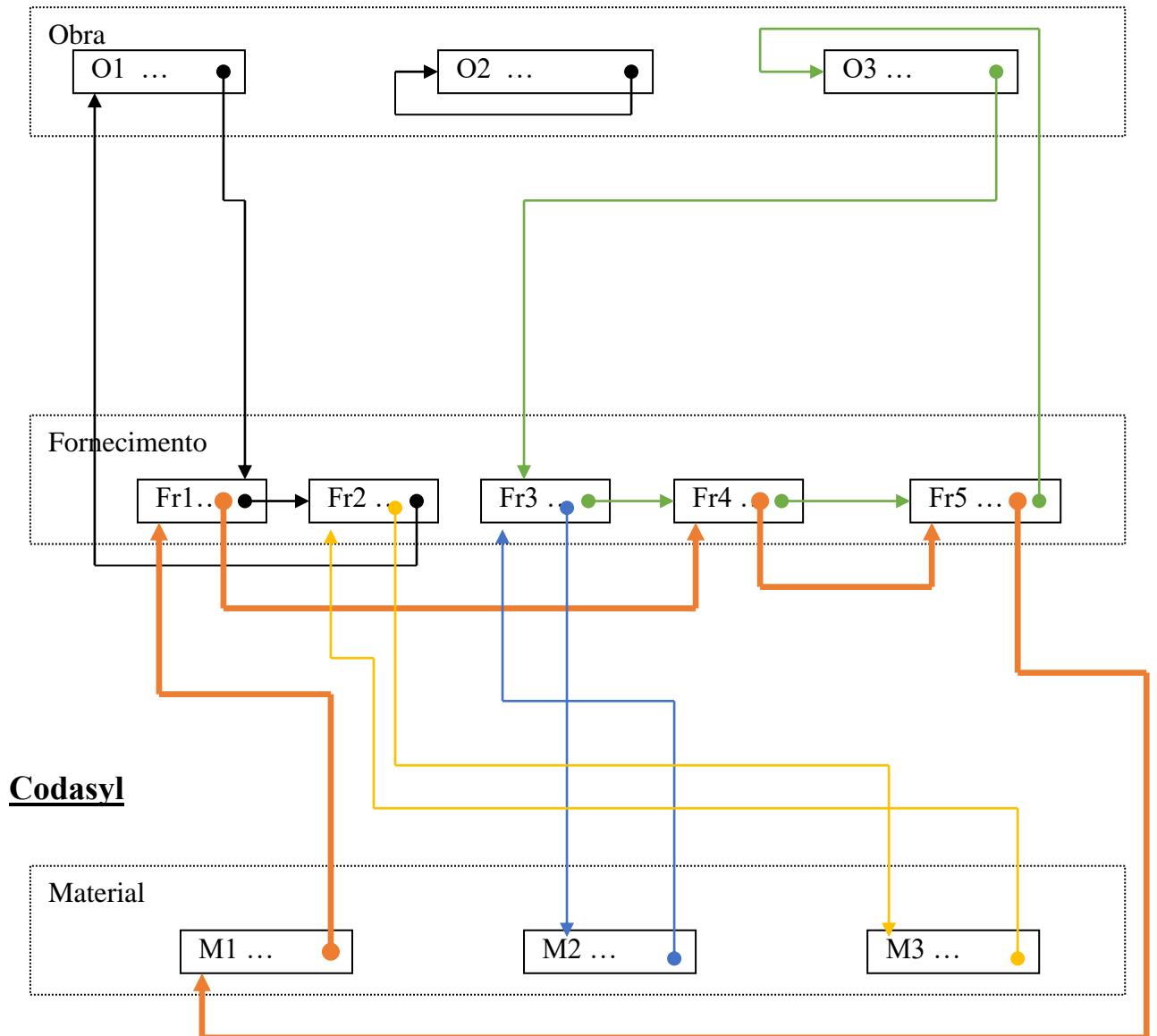
Esquema da base de dados:

(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 procedural (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

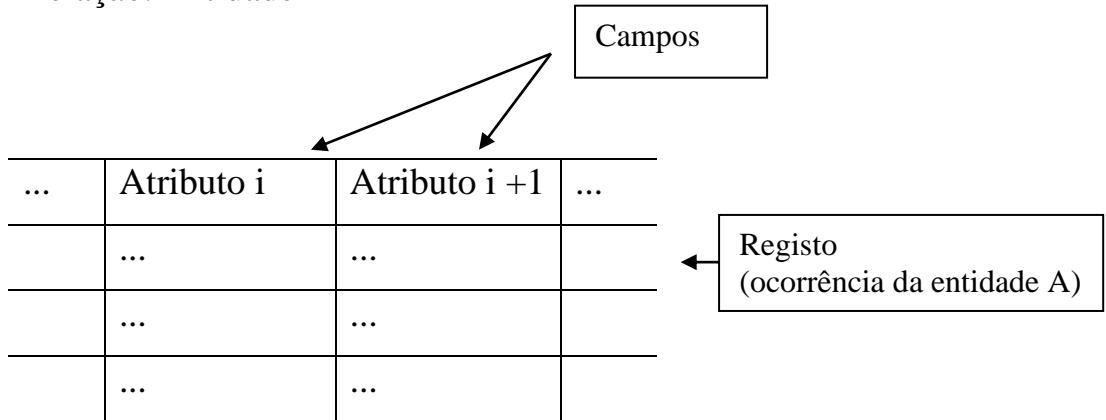
Fornecimento

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 relações (tabelas).

Relação: Entidade A



Terminologia

Formal	Implementação	Analogia
Relação	Tabela	Ficheiro
Tuplo	Linha	Registo
Atributo	Coluna	Campo

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 procedural (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 selecionar 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).

- . Seleciona 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: ?

- **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 respetivas 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

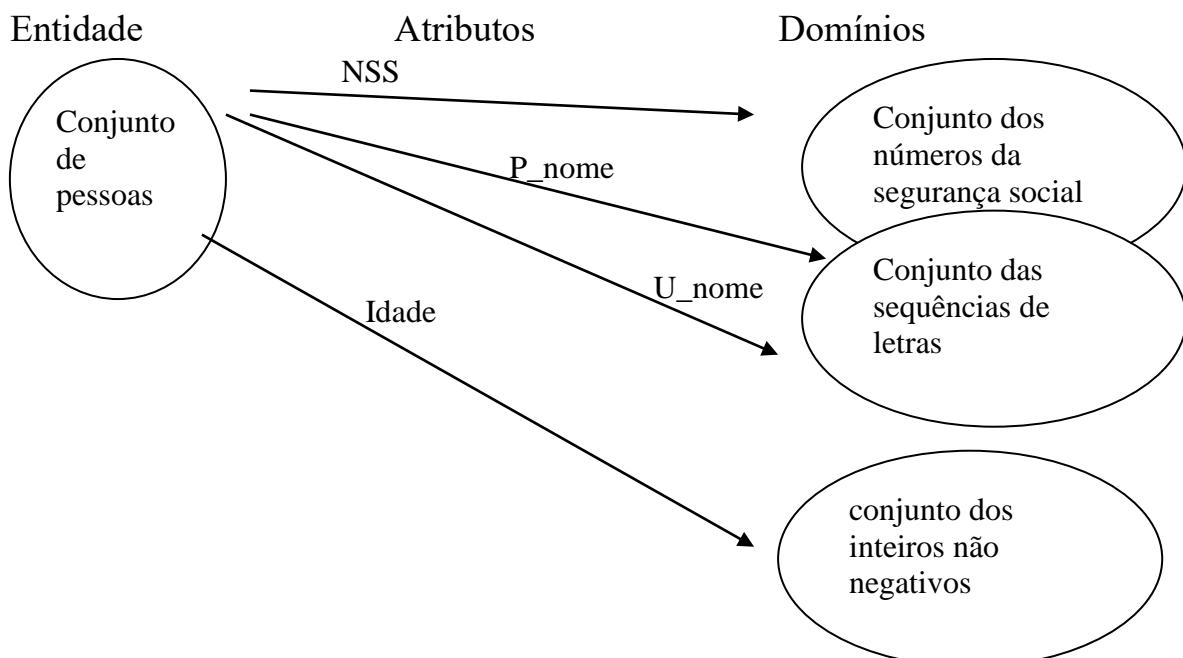
$Pessoa(NSS, P_nome, U_nome, Idade)$ é interpretada como a função

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

Esta função determina para cada pessoa p um 4-uplo (n° 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*

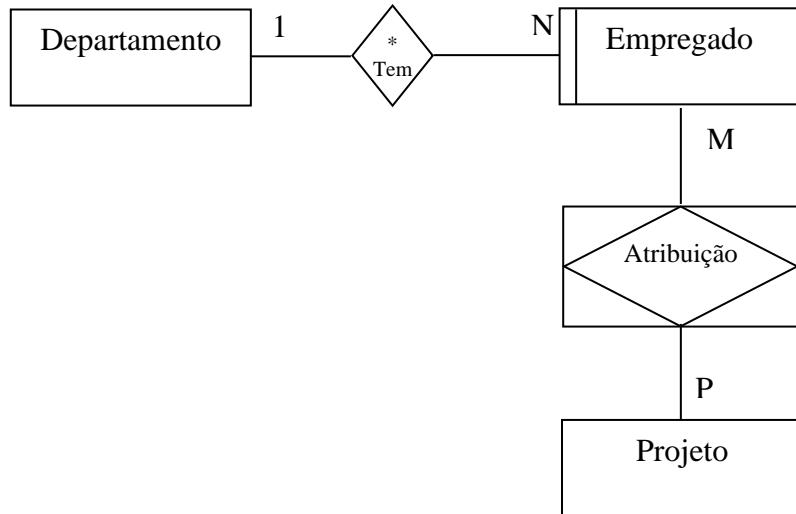
Emp# – número de segurança social do empregado
Proj# – código do projeto
Funcao – função que o empregado desempenha no projeto

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 a 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

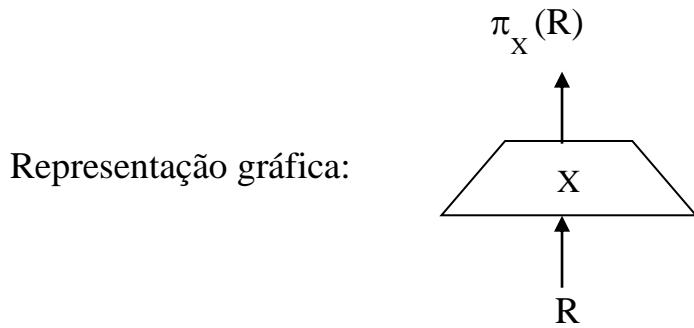
Seja $R(X, Y)$ com $X = A_1, A_2, \dots, A_k$

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

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.



Exemplo:

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

. Emp# é chave da relação empregado

Empregado

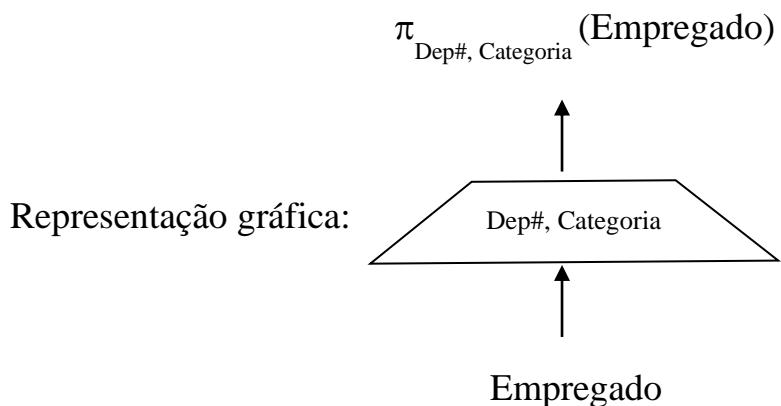
Emp#	Nome	Categoria	Dep#
e1	n1	c1	d1
e2	n2	c2	d2
e3	n3	c3	d1
e4	n4	c1	d2
e5	n5	c2	d3
e6	n6	c2	d3
e7	n7	c1	d1

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

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

dá origem à tabela:

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



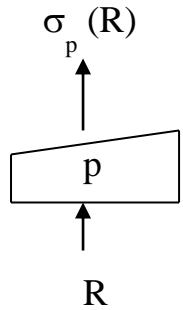
2.2.2.2 Restrição (ou seleção)

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 .

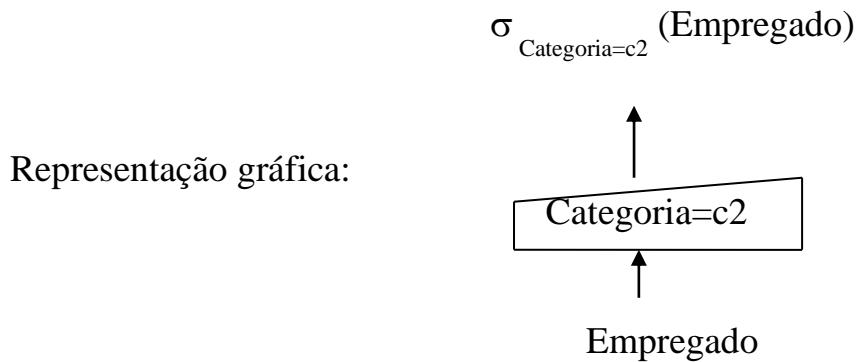


Exemplo: Restrição da tabela *Empregado* tal que *Categoria=c2*

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

dá origem à tabela,

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



2.2.2.3 Operações com conjuntos

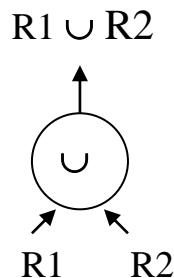
União, Intersecção e Diferença

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.

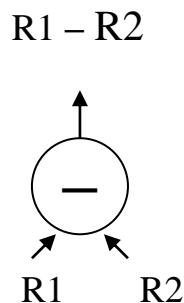
2.2.2.3.1 União

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



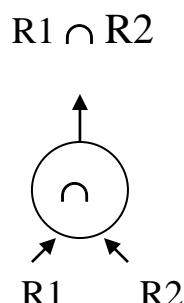
2.2.2.3.2 Diferença

$R1 - R2$ 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 $R2$.



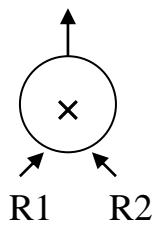
2.2.2.3.4 Produto Cartesiano

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 $R2$.

$R1 \times R2$



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 :

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

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:

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

denota a composição de duas operações:

- A junção natural das relações *Empregado* e *Departamento* sobre os atributos *Dep#*;
- A projeção 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

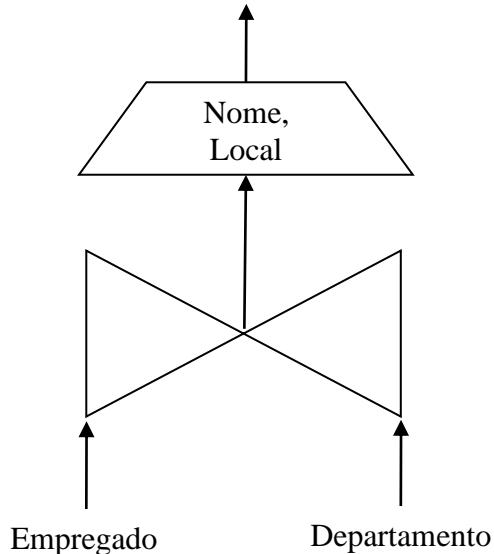
Dep#	Nome	Local
d1	N1	l1
d2	N2	l1
d3	N3	l2

Resultado:

Nome	Local
n1	11
n2	11
n3	11
n4	11
n5	12
n6	12
n7	11

Representação gráfica:

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



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:

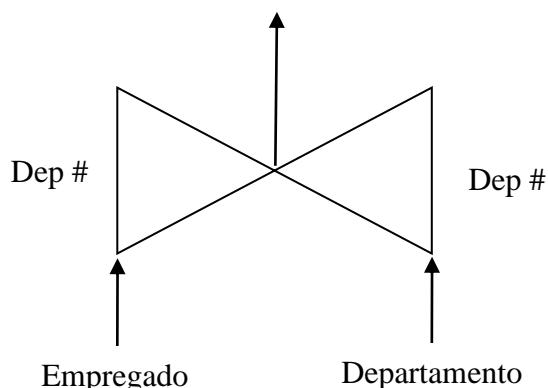
Obtenha todos os pormenores dos empregados e dos seus departamentos.

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

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

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



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 , respectivamente.

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

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 , $R \bowtie L S$, 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.

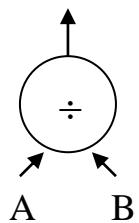
A divisão de A por B sobre Y e Z é

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

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

Representação gráfica:

$$A \div B$$



Exemplo: 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 $\pi_{\text{Proj}\#}(\text{Projeto})$ 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 ÷ d1; b) D ÷ d2; c) D ÷ 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 expressão E , e opcionalmente designa os seus atributos por A_1, A_2, \dots, A_n .
ou

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	
		SUM (B) = $2+4+2+2 = 10$
1	2	AVG (A) = $(1+3+1+1)/4 = 1.5$
3	4	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.

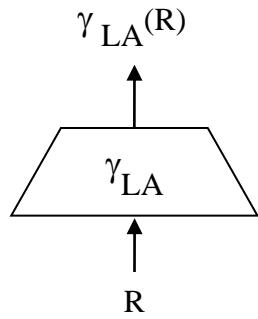
O operador γ tem um índice, uma lista L . Cada elemento de L é um:

- 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?

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

$$\frac{\text{NEmps}}{3}$$

ii) Quantos empregados tem cada departamento?

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

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 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 .

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

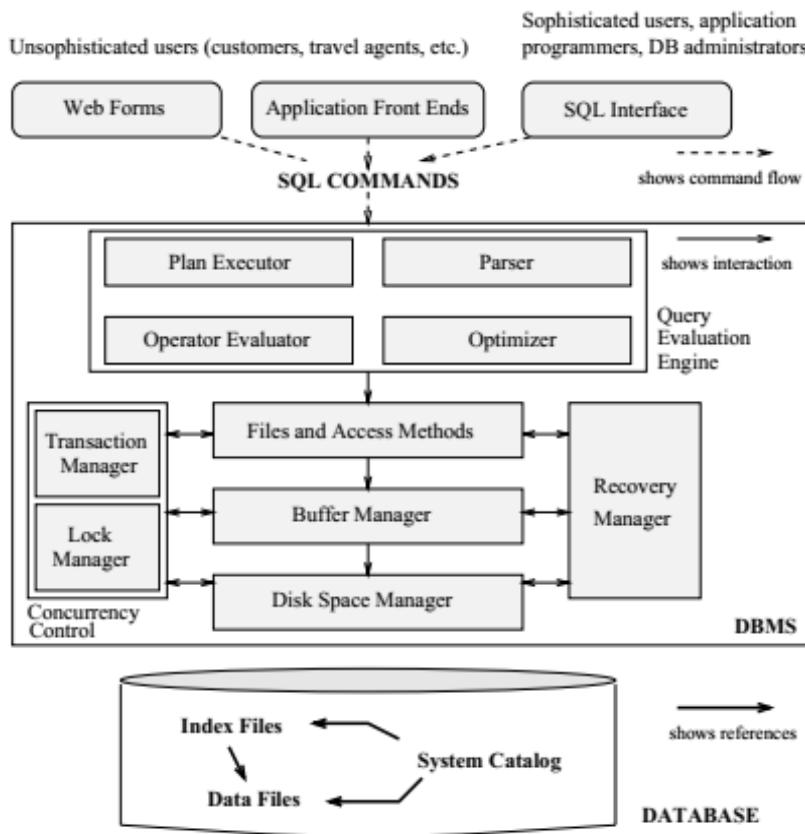
$R \bowtie_F S = \pi_L(\sigma_F(R \times S))$, onde L é a lista formada pelos atributos de R seguidos dos atributos de S que não estão em R .

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 optimizador 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 registo. 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

registro (*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 Transacções (*transaction manager*), que garante que as transacções solicitam e libertam os trincos segundo um protocolo de bloqueio apropriado e agenda a execução das transacções; o Gestor de Trincos (*lock manager*), que mantém um registo de pedidos e de concessões de trincos sobre os objetos da base de dados e de quando estes ficam disponíveis; e o Gestor de Recuperações (*recovery manager*), que é responsável por manter o *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 aspecto importante do processamento de consultas é a otimização. Como podem existir várias transformações equivalentes da mesma consulta de alto nível, o objetivo do optimizador 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 *privilegios*.

Um aspeto interessante dos SGBD 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.

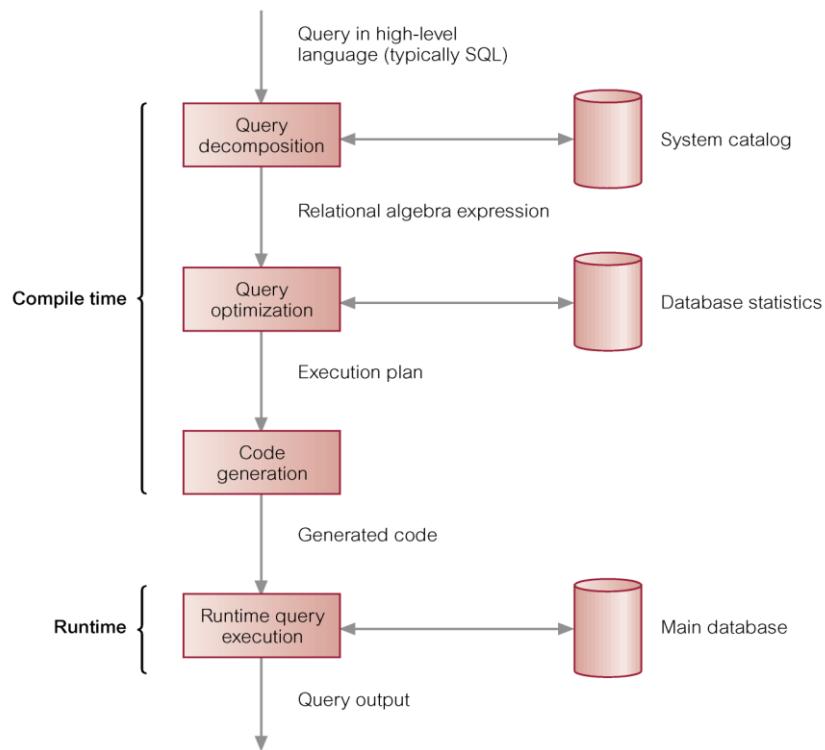
Os objetivos da decomposição de consultas são transformar uma consulta de alto nível numa consulta de álgebra relacional e verificar se a consulta é sintática e semanticamente correta. Os estágios típicos da decomposição são: análise, normalização, análise semântica e reestruturação da consulta.

- *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 satisfeita 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 das quais também podem introduzir redundância. Se o utilizador não tiver o acesso apropriado a todos os componentes da consulta, a consulta é rejeitada.

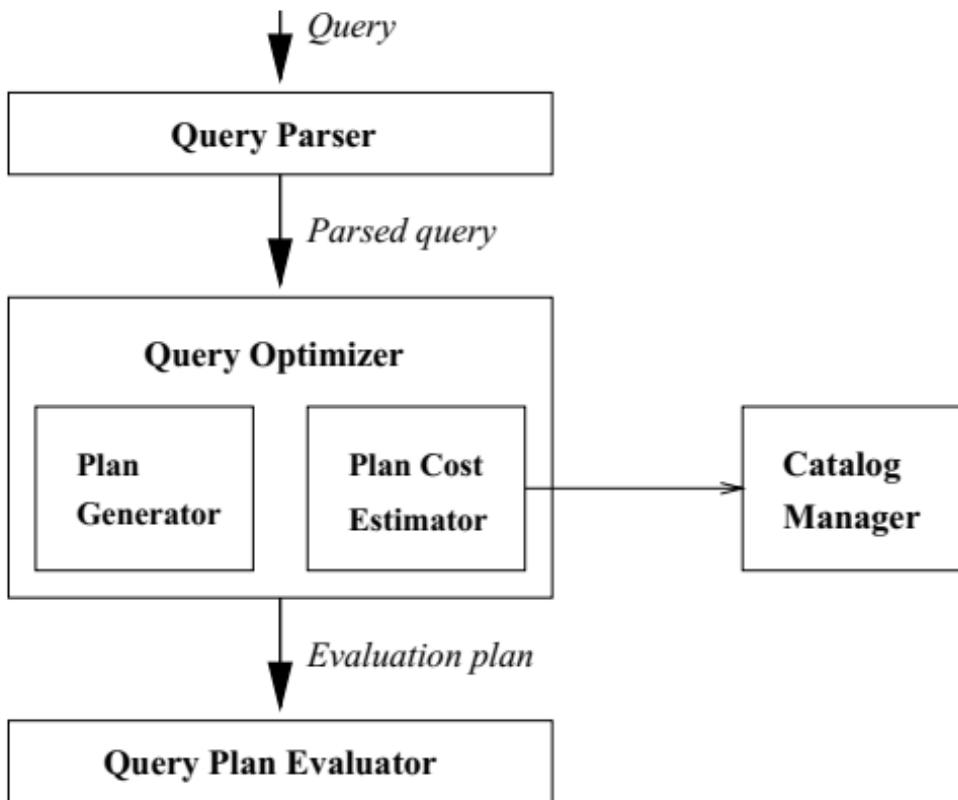
- *Reestruturação da consulta* – na etapa final da decomposição, a consulta é reestruturada para fornecer uma implementação mais eficiente.

2.3.5 Otimização de consultas¹⁰

O objetivo do optimizador de consultas é encontrar um bom plano de avaliação para uma determinada consulta. O espaço de planos considerado por um optimizador 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 optimizador 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 se implementação a ser usado para cada operador relacional.

Considerando a consulta seguinte:

```

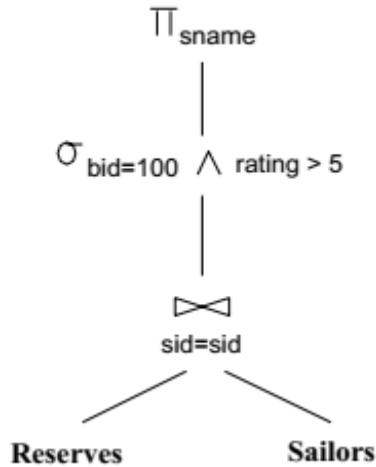
SELECT S.sname
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
AND R.bid = 100 AND S.rating > 5

```

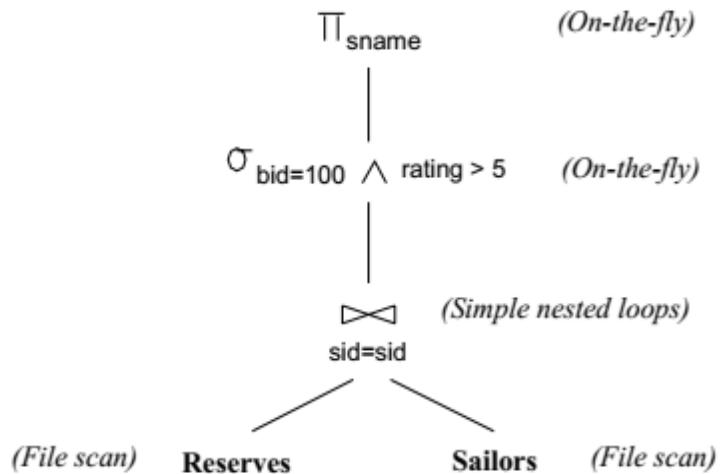
que se pode exprimir em álgebra relacional da forma seguinte:

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

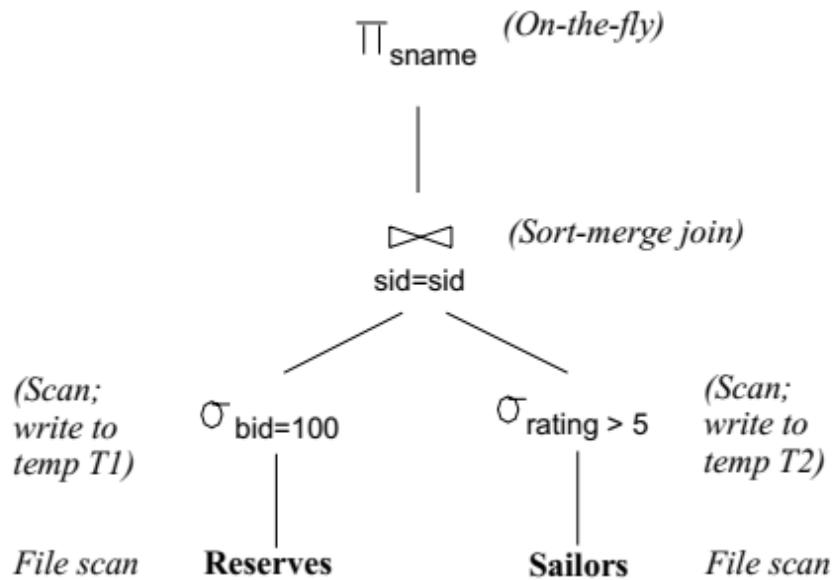
Em termos de representação gráfica (árvore):



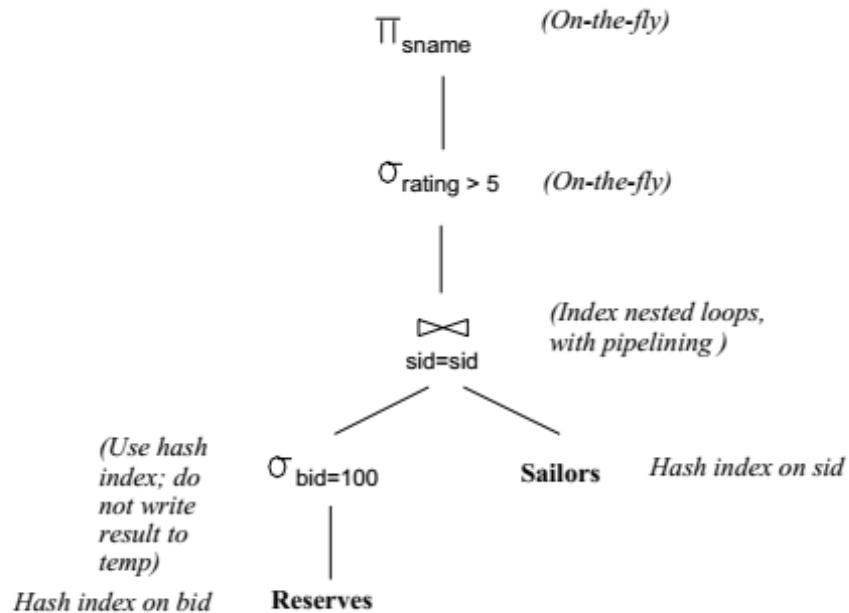
Plano de avaliação:



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. 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 procedural, 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 selecionar 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)

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

- 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 selecionar 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 * → Seleciona todos os atributos.

- *Selecção, 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.*

{ $\sigma_{DepNum = 7 \vee DepNum = 9}$ (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.*

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

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:

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

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.

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

```
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 selecionados 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

Annotations:

- Above row 1: "Não pode ser negativo" (Cannot be negative) points to the Salário cell.
- To the left of row 4: "O campo não pode ser nulo" (The field cannot be null) points to the Nome cell.
- To the right of row 4: "Demasiado pequeno" (Too small) points to the Salário cell.
- To the right of row 4: "Data inválida" (Invalid date) points to the Data_Nasc cell.

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

Emp#	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

Emp#	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

Dep#	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

NumEmp	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.

Questão:

Armazenar dados de forma redundante, i.e., em mais do que um lugar da base de dados, pode levar a vários problemas.
Indique o que se entende por anomalias de inserção, de eliminação e de modificação.

Resposta:

Anomalia de inserção: pode não ser possível inserir certa informação a menos que outra informação, não relacionada, seja também armazenada.

Anomalia de eliminação: pode não ser possível eliminar certos dados sem perder também outros dados não relacionados.

Anomalia de modificação: se uma cópia dos dados redundantes é actualizada, é criada uma inconsistência a menos que todas as cópias sejam similarmente actualizadas.

Podemos evitar as anomalias anteriores se decompormos 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:

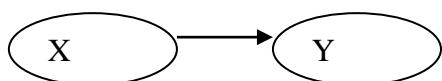
- . Dependências Funcionais
- . Dependências Multivalor
- . Dependências de Junção

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) se e só se **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

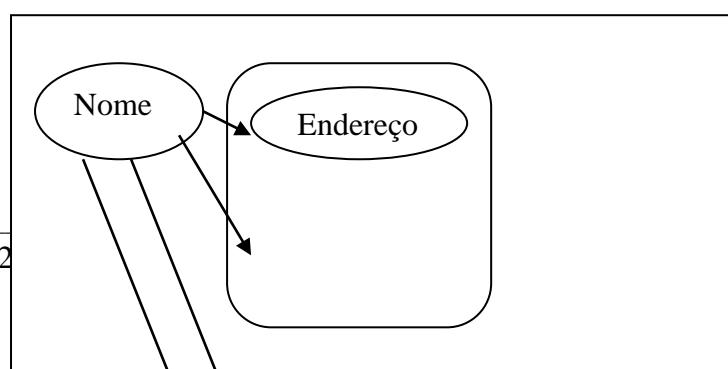


- . X determina Y
- . Y depende de X

(1) Exemplo

$Nome \rightarrow Endereço$

$Nome \rightarrow Cidade$

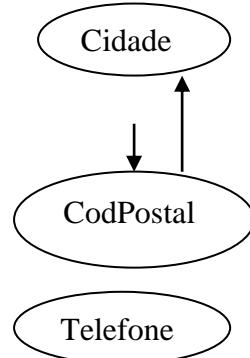


$Nome \rightarrow CodPostal$

$Nome \rightarrow Telefone$

$Endereço, Cidade \rightarrow CodPostal$

$CodPostal \rightarrow Cidade$



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

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

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 → Endereço

Nome → Cidade

Nome → CodPostal

Nome → Telefone

Endereço, Cidade → CodPostal

CodPostal → Cidade

A – Por reflexibilidade,

Endereço, Cidade → Endereço

Endereço, Cidade → Cidade [1]

B – De “*Endereço, Cidade → CodPostal*” e “*CodPostal → Cidade*” ,

por transitividade obtemos

Endereço, Cidade → 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}

$$b) XW \rightarrow YW \wedge YW \rightarrow Z \Rightarrow XW \rightarrow Z \quad \{a), \text{ hipótese, transitividade}\}$$

Exemplo:

Aplicando a pseudotransitividade às dependências funcionais

$Nome \rightarrow Endereço$ e $Endereço, Cidade \rightarrow CodPostal$

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 *implicada* 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 $BC \rightarrow AD$ 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 **irreduzível** sse satisfaz as seguintes propriedades:

- O lado direito (dependente) de qualquer DF em S contém apenas um atributo (i.e., é um conjunto singular);

¹³ (Date, 2004), pág. 341.

- 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).

Para todo o conjunto de DFs existe pelo menos um conjunto equivalente que é irredutível. Pode existir mais do que uma cobertura irredutível (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 irredutível e equivalente a F .

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

$$\begin{aligned} A &\rightarrow B \\ A &\rightarrow C \\ B &\rightarrow C \\ A &\rightarrow B \\ AB &\rightarrow C \\ AC &\rightarrow D \end{aligned}$$

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 irredutível.

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

$$R (Nome, Telefone, Cidade)$$

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

Nome \rightarrow *Telefone*

Nome \rightarrow *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) $R(X, Y, Z) \supseteq \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$:

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, $R(X, Y, Z) = \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$. **c.q.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:

1^a Forma Normal

2^a Forma Normal

3^a Forma Normal

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

Forma Normal de Boyce-Codd (FNBC)

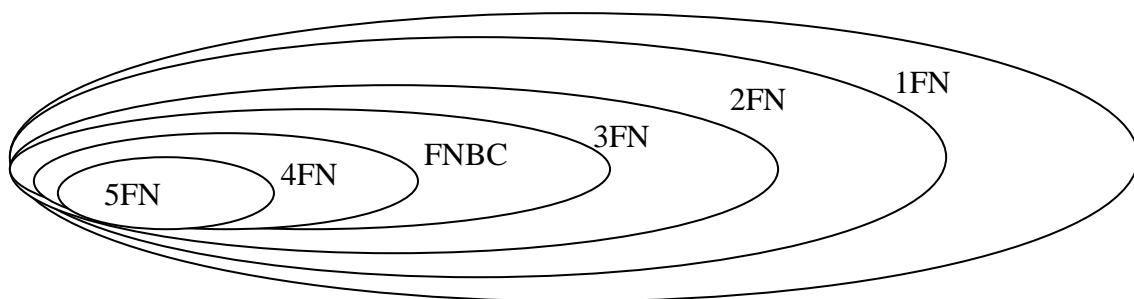
Mais tarde Fagin (1979) propôs:

4^a Forma Normal

5^a 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 **1^a Forma Normal** se

- . Cada atributo contém apenas valores atómicos.
- . 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 atómicos!!!**

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:

$Fornecedor \rightarrow Peça$

$Peça \rightarrow Fornecedor$

$Fornecedor \rightarrow Cidade$

Num dado instante t ,

R

<u>Fornecedor</u>	<u>Peça</u>	<u>Cidade</u>	<u>Quantidade</u>
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

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

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

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

$$Cod_produto \rightarrow Desc_produto \text{ (1)}$$

$$Cod_produto \rightarrow Preço_produto \text{ (2)}$$

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 $E(N_emp, Dep, Cidade)$, onde:

$$N_emp \rightarrow Dep$$

$$Dep \not\rightarrow N_emp$$

$$Dep \rightarrow Cidade$$

$$Cidade \not\rightarrow Dep$$

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 (N_emp, Dep, Cidade) está em 2FN!

Se substituirmos *E* por:

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

e

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

desaparecem as anomalias.

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

Ficamos com o esquema relacional:

Empregado (N emp, Dep)

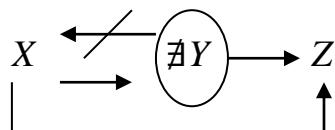
Departamento (Dep, Cidade)

3.3.3.1 DF Direta

Definição

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

$\nexists_{Y \in R} : Y \not\rightarrow X, X \rightarrow Y$ e $Y \rightarrow Z$ (não trivial).



$X \rightarrow Z$ é direta

3.3.3.2 Terceira Forma Normal (3FN)

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

R está na 3^a Forma Normal sse

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

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 $X \rightarrow Y$, $Y \not\rightarrow X$ 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:

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

$Cod_cliente \rightarrow Nome_cliente$

$Cod_cliente \rightarrow Morada_cliente$

por união, $Cod_cliente \rightarrow Nome_cliente, Morada_cliente$

(logo a DF $N_nota_enc \rightarrow Nome_cliente, Morada_cliente$ (não é directa))

A relação não está na 3FN.

Decomposição:

$Cliente(Cod_cliente, Nome_cliente, Morada_cliente)$

$Nota_enc(N_nota_enc, Cod_cliente)$

Exemplo 2:

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

onde, $Código \rightarrow Nome$ }
 $Nome \rightarrow Código$ } \Rightarrow (por transitividade) $Nome \rightarrow Morada$
 $Código \rightarrow Morada$ }

$Código \rightarrow Morada$ - Esta dependência funcional é direta?

A relação está na 3^a forma normal?

Resposta: _____

Exercício: Normalize em 3FN o esquema relacional abaixo.

*Clientes (N_cliente, (Endereços_para_remessas) *, 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

$Cidade, Endereço \rightarrow Cod_postal$

$Cod_postal \rightarrow Cidade$

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 $\text{Cod_postal} \rightarrow \text{Cidade}$

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

Decomposição:

$R1 (\underline{\text{Cod_postal}}, \text{Cidade})$

$R2 (\underline{\text{Endereço}}, \underline{\text{Cod_postal}})$

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

Exemplo 1:

$E (\underline{\text{Emp}}, \text{Dep}, \text{Cidade})$

$\text{Emp} \rightarrow \text{Dep}$

$\text{Dep} \not\rightarrow \text{Emp}$

$\text{Dep} \rightarrow \text{Cidade}$

$\text{Cidade} \not\rightarrow \text{Dep}$

Não está na 3FN (porque a DF $\text{Emp} \rightarrow \text{Cidade}$ não é direta)

Não está na FNBC (porque na DF $\text{Dep} \rightarrow \text{Cidade}$ 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.

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

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

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

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

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

Chaves candidatas:

N_{forn}

$Nome_{forn}$

3 FN? _____

FNBC? _____

Exemplo 4: Relação com duas Chaves candidatas sobrepostas.

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

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

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

$$N_{forn}, N_{Peça} \rightarrow Qtd$$

Chaves candidatas:

$N_{forn}, N_{peça}$

$Nome_{forn}, N_{peça}$

3FN? sim

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

N_{forn} não é chave candidata.

Decompor em:

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

$$F2(N_{forn}, N_{peça}, Qtd)$$

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 $X \rightarrow Y$, $Y \not\rightarrow X$ 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 $X \not\rightarrow Z$ (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 Modelação de dados (DEA)

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 Registar 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 Normalização¹⁵.

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.

¹⁵ Codd, E. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of ACM, 13(6).

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 Chen¹⁶ 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 relacional¹⁷.

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

¹⁶ Chen, P. (1976). The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1).

¹⁷ (Coronel & Morris, 2016), página 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 atómico (indivisível) de informação.

Exemplos: Nº 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

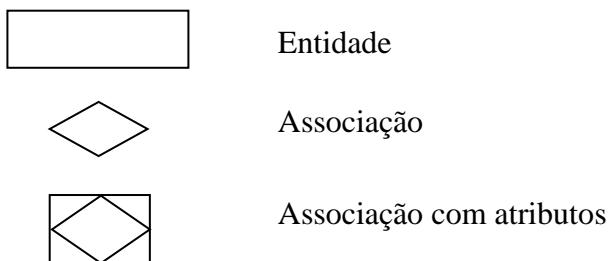
Diagrama Entidade/Associação:

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.



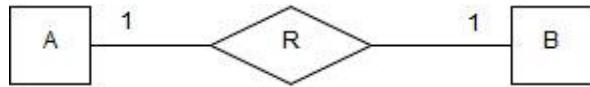
4.3 Propriedades das associações

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:

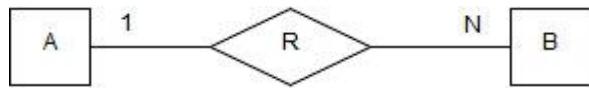


Exemplo:

Seja um curso em que cada módulo é assegurado por um monitor e cada monitor assegura apenas um módulo

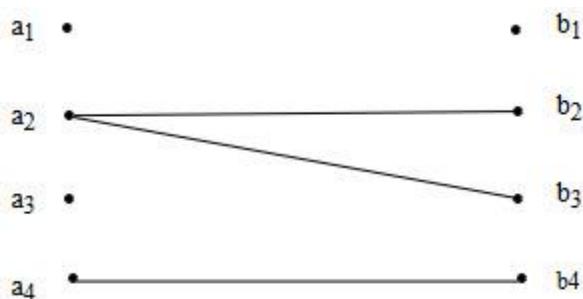


4.3.1.2 Associação 1:N



- 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).

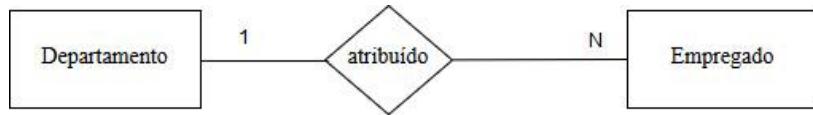
Diagrama de ocorrências:



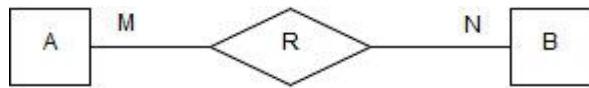
Exemplo:

Um departamento tem atribuídos vários empregados (eventualmente só um, ou mesmo nenhum).

Um empregado está atribuído a apenas um departamento (ou nenhum).

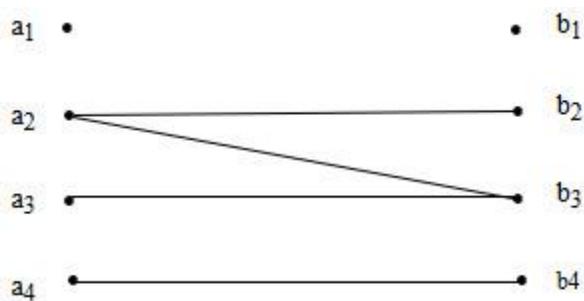


4.3.1.3 Associação M:N



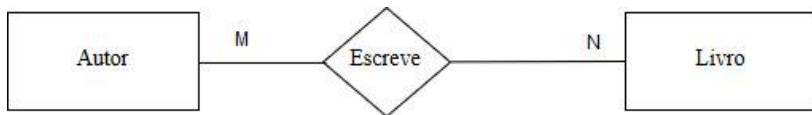
- A cada ocorrência da entidade *A* está associada uma, várias ou nenhuma ocorrência da entidade *B*.
- A cada ocorrência da entidade *B* está associada uma, várias ou nenhuma ocorrência da entidade *A*.

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 registo 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 registo 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 registo.

4.3.2 Tipo de participação

Uma entidade pode participar numa associação de duas formas:

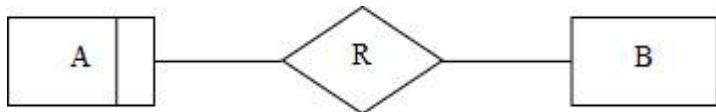
- **Obrigatória**

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:

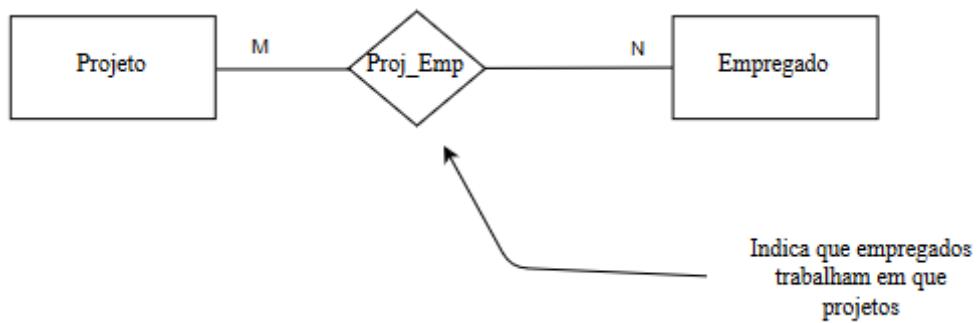
- Um departamento tem de ter pelo menos um empregado
- Um empregado tem de pertencer a um departamento

4.4 Decomposição de Associações $M:N$

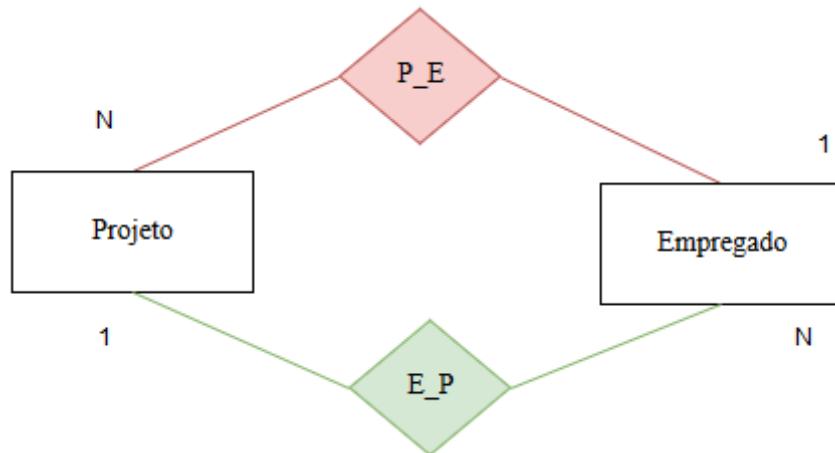
Nas fases iniciais do processo de modelação podem surgir associações muitos-para-muitos ($M:N$). No entanto, na fase final da modelação de dados não devem aparecer relações $M:N$ pois não são diretamente suportadas pelos SGBDs, levando à criação de uma nova **entidade associativa** ou de intersecção.

Qualquer associação $M:N$ entre dois tipos de entidades pode ser decomposta em duas associações $1:N$.

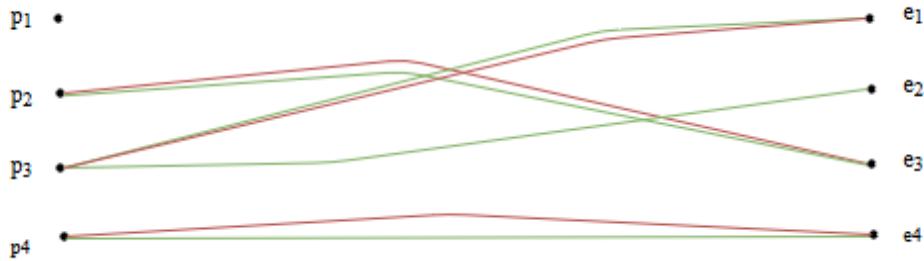
Seja a associação:



Note-se que este DEA é diferente do seguinte:

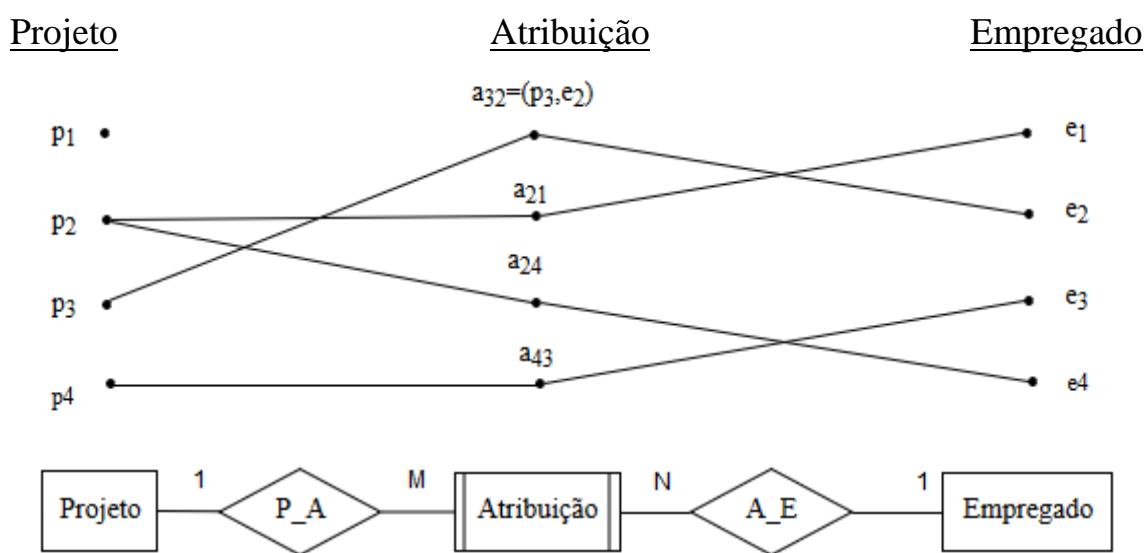


Repare-se no diagrama de ocorrências:



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

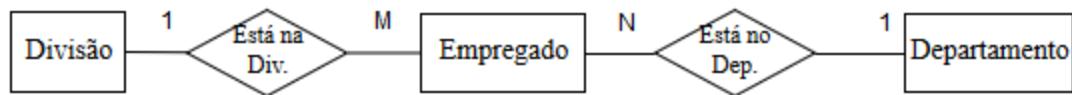
a) Desenhar um diagrama E-A mostrando uma associação entre as entidades Fornecedor e Peça

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.
- Desenhar um diagrama E-A mostrando as entidades *Professor*, *Curso* e *Sala* e as associações *Prof_Curso* e *Sala_Curso*.
 - 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?
- Qualquer associação *M:N* pode ser decomposta em duas associações *1:N*.
 - 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.

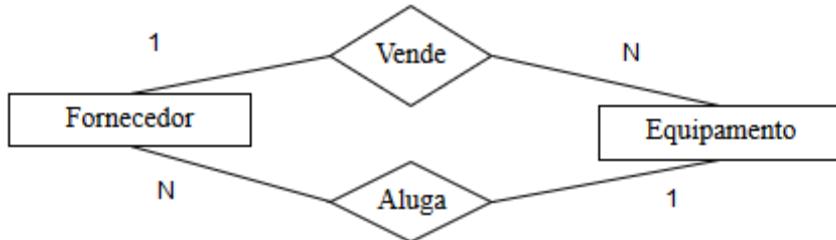
Possuir uma dada habilitação permite ao empregado manter vários tipos de máquinas apesar de a manutenção de um tipo de máquina requerer uma habilitação específica.

- 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_Operation	Habilitação
Habilitação	Permite_Manutenção	Tipo_Máquina

- b) Redesenhe o diagrama com as associações *Necessita_para_Operation* e *Permite_Manutenção* combinados numa só associação *M:N, Operation_ou_Manutenção*.
- c) Redesenhe o diagrama anterior com a associação *Operation_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			

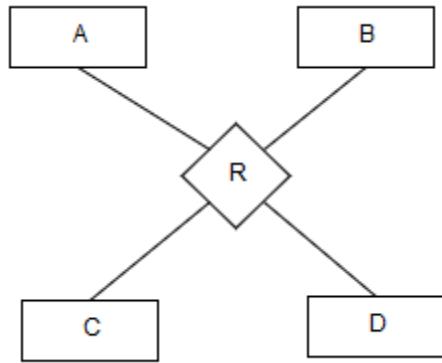
- a) Crie uma tabela de ocorrências que mostre que equipamento cada fornecedor vende **ou** aluga.

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

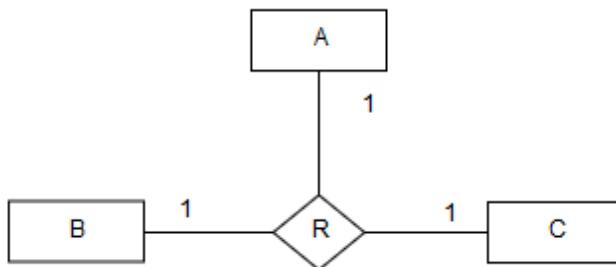
Relacionam entre si mais do que duas entidades.



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.

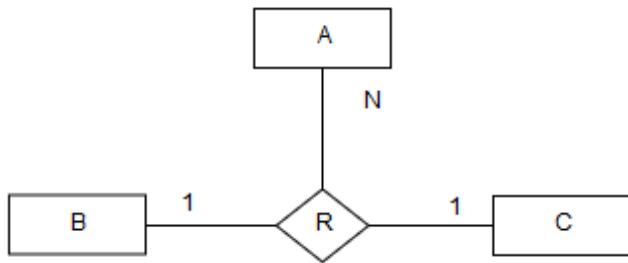
4.5.1 Grau 1:1:1



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).

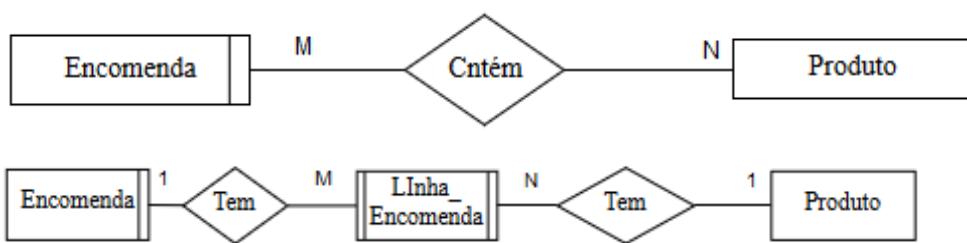
Análogo para,

Grau $1:M:N$

Grau $M:N:P$

Vimos que:

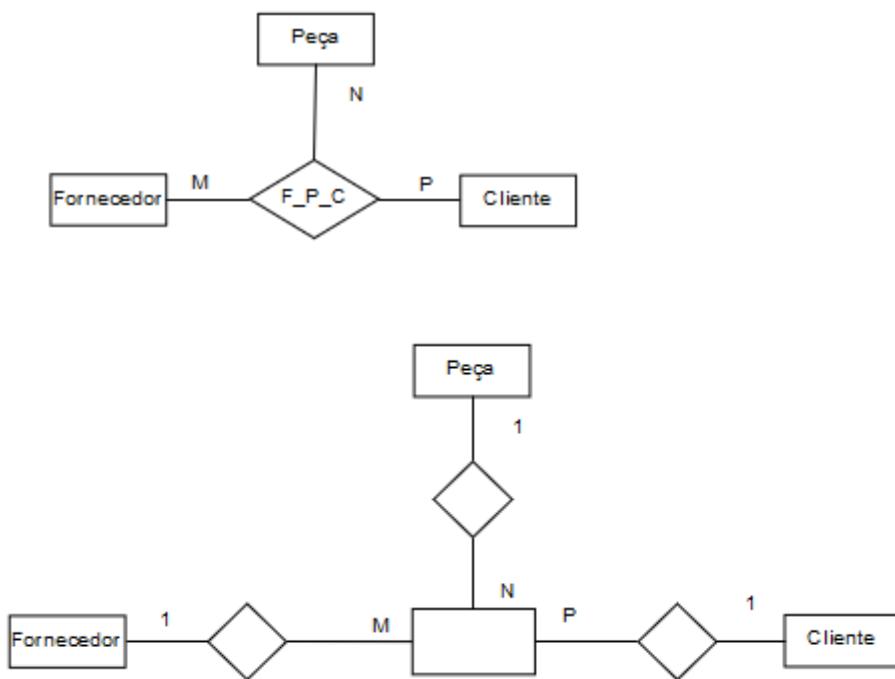
As associações $M:N$ devem ser substituídas por um par de associações de grau $1:N$.



A decomposição permite:

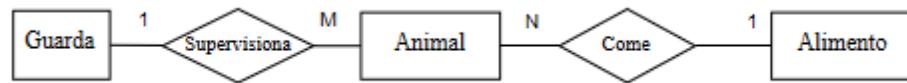
- Ressaltar a existência de entidades não identificadas no início;
- (Facilita) a análise do diagrama em termos de consistência;
- Dar ao modelo a forma adequada para passos subsequentes do método.

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 real¹⁸.

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.

¹⁸(Connolly & Begg, 2015), pág. 426.

Exemplo 1:



Permite responder às questões:

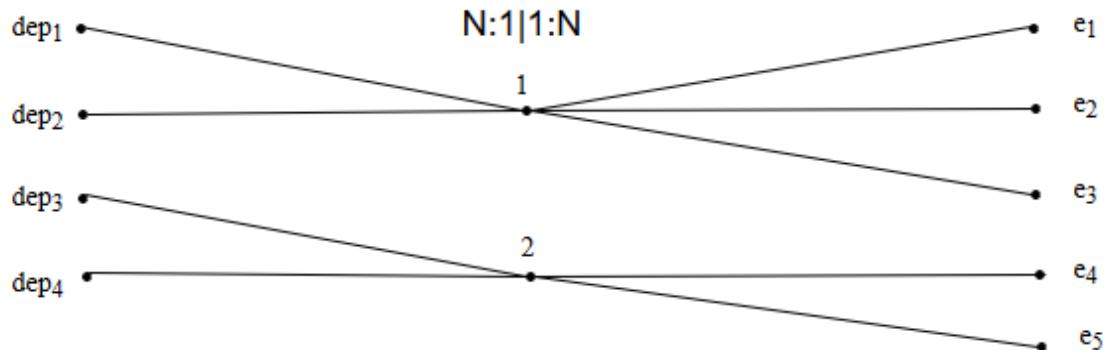
- Quais os empregados de uma direção?
- Quais os departamentos de uma direção?
- A que direção pertence um empregado?
- A que direção pertence um departamento?

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 ?)

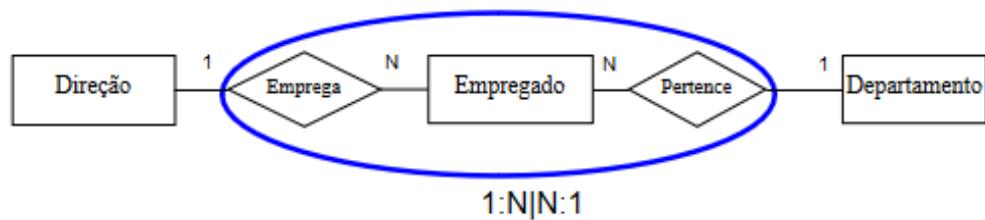
Uma possibilidade de remover o *fan trap* e responder às questões, é corrigir as associações e reestruturar o DEA original:

Tentativa 1:



Problema: *Mas que fazer se um empregado pertencer a uma direção sem pertencer a nenhum dos seus departamentos?*

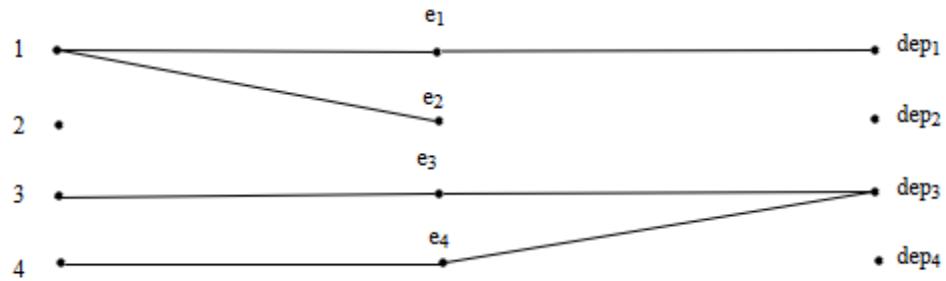
Tentativa 2:



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*.

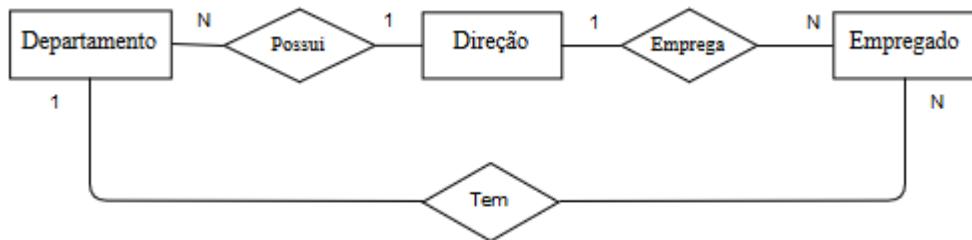
Chasm trap – Where a model suggests the existence of a relationship between entity types, but the pathway does not exist between certain entity occurrences.



(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

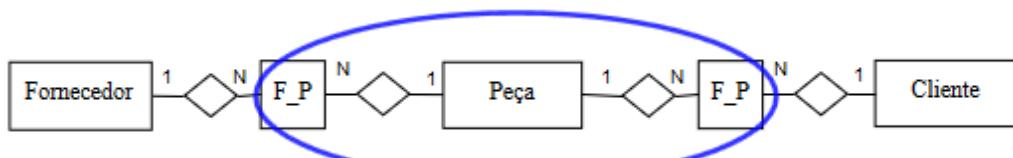
P1 é fornecida a C2

F2 fornece P2

P2 é fornecida a C1

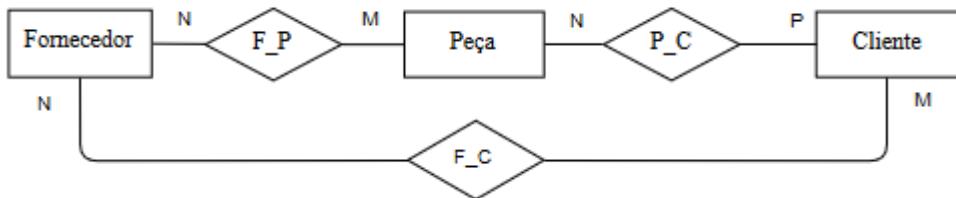
Problema: A pergunta “que fornecedores fornecem um dado cliente e o quê?” não pode ser respondida!

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:

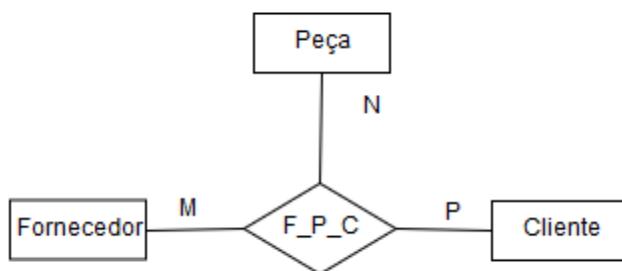
- Que fornecedores fornecem que peças?
- Que peças são fornecidas a que clientes?
- Que fornecedores fornecem que clientes?

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 $N:I|I:N$.

Exercício: decompondo as associações M:N.

Tentativa 3 (solução):

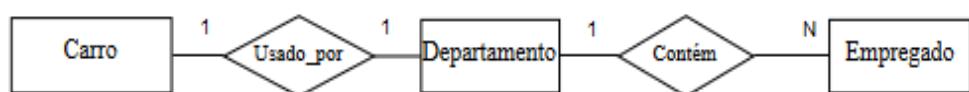


Como se viu anteriormente, as associações $M:N$ devem ser decompostas.

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.

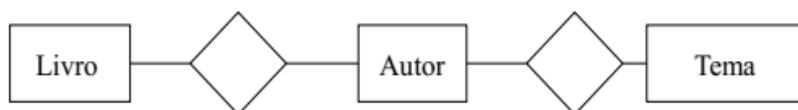
Proponha um diagrama E-A que contenha apenas duas associações.

5. Um modelo conceptual representa autores e a classificação por temas dos seus livros.

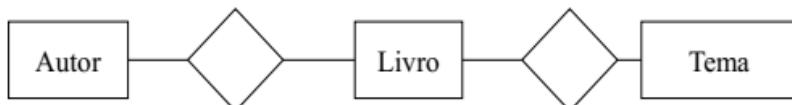
Não há dependências funcionais entre as chaves das entidades, *Autor, Livro, Tema*, mas um autor está sempre associado com todas as classificações aplicadas ao livro que escreveu.

Discuta as vantagens das estruturas seguintes. Qual escolheria?

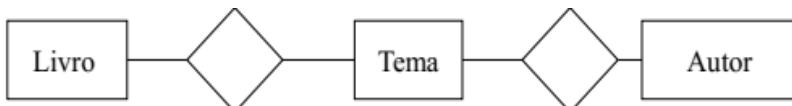
a)



b)



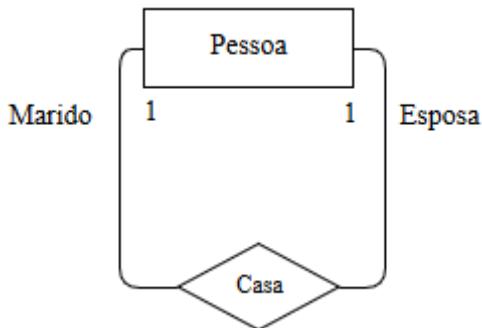
c)



4.7 Associações Unárias

4.7.1 Associações 1:1

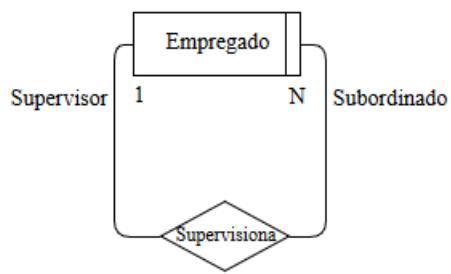
Considere-se os casamentos numa sociedade onde uma do género masculino pode casar com uma pessoa do género feminino.



A entidade *Pessoa* tem “dois papéis”: *Marido* e *Esposa*.

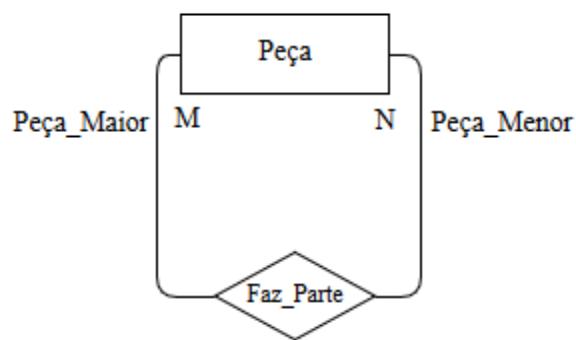
4.7.2 Associações 1:N

- 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).



4.7.3 Associações M:N

A fabricação de peças é feita a partir de outras peças.



5 Modelação (modelo lógico)

5.1 Esquema Relacional

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.

5.1.1 Associações 1:1

Suponhamos que os carros de uma companhia são atribuídos numa relação de 1 para 1:

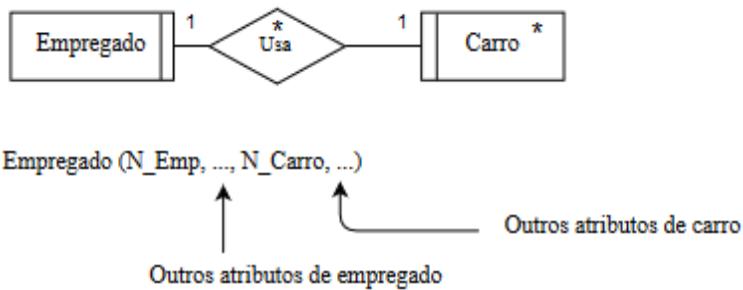
- Nenhum carro é partilhado entre empregados
- Nenhum empregado utiliza mais do que um carro



Caso 1: As duas entidades são obrigatórias

- Todo o carro é usado por um empregado
- Todo o empregado tem um carro da companhia

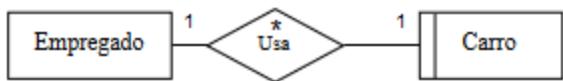
Basta uma tabela para representar a situação. Os atributos de carro podem ser vistos como atributos adicionais de empregado.



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)

5.1.2 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, ...)

Contém (N_Paciente, Nome_Enf)

5.1.3 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.

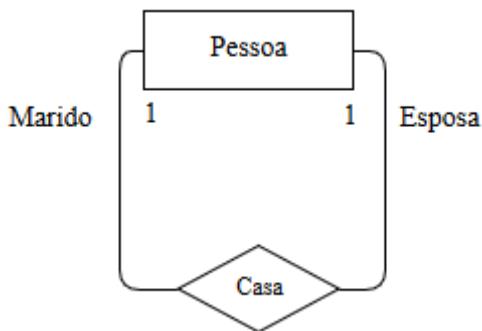
Uma associação sem atributos dá origem a uma tabela quando:

- 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 $M:N$.

5.2 Associações Unárias

5.2.1 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, ...)

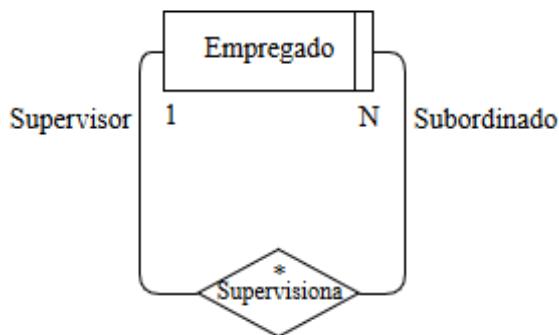
Casa (BI_Esposa, BI_Marido)

Exercício:

- Considere a possibilidade de dissolução do casamento (divórcio) e de novo casamento.*
- Considere o casamento entre pessoas do mesmo género.*

5.2.2 Associações 1:N

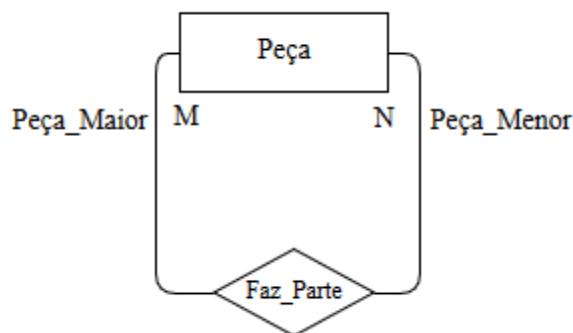
- Um empregado pode ser supervisor e/ ou subordinado;
- Alguns empregados não supervisionam outros, mas todo o empregado tem um supervisor (o empregado mais antigo supervisiona-se a si próprio).



Empregado (N_Emp, Nome, ..., N_Emp_Chefe)

5.2.3 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 nº 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*?

5.3 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.

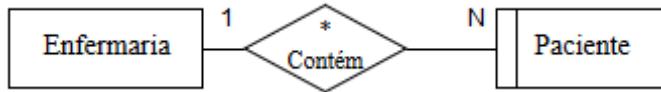
Cada exemplar é identificado por um número de exemplar e cada sócio por um número de sócio. Outros atributos são *Título*, *Data_de_aquisição*, *Preço_de_aquisição*, *Data_empréstimo*, *Nome_sócio*, *Limite_sócio*.

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.

5.4 Extensão do esquema relacional

Considere-se o modelo:



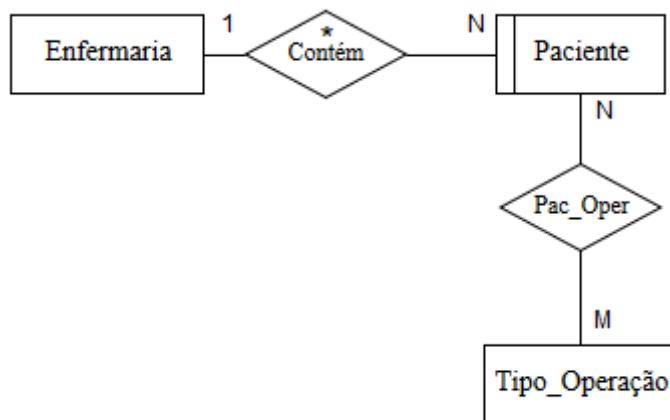
Enfermaria (Nome_Enf, Tipo_Enf, N_de_Camas)

*Paciente (N_Paciente, Nome, Data_Nascimento,
Data_Internamento, Nome_Enf)*

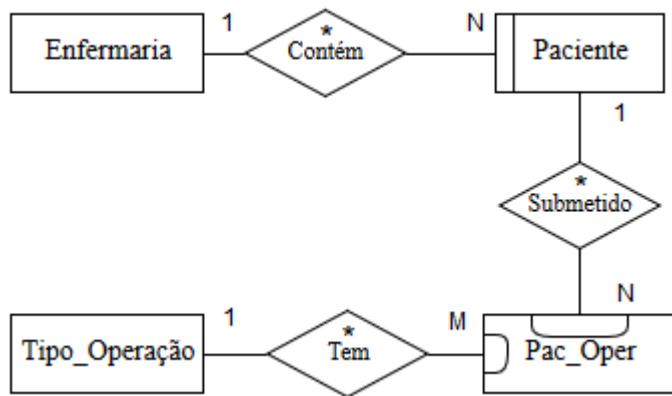
Suponha que é necessário incluir os atributos código de operação e nome de operação, onde código de operação determina nome de operação.

- Um paciente pode submeter-se a várias operações.

Redefina o modelo.



Decompondo a associação $M:N$:



Enfermaria (Nome_Enf, Tipo_Enf, N_de_Camas)

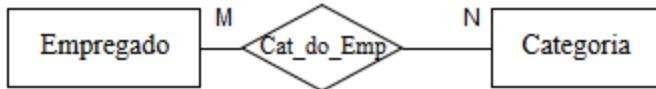
*Paciente (N_Paciente, Nome, Data_Nascimento,
Data_Internamento, Nome_Enf)*

Tipo_Operação (Cod_Operação, Designação)

Pac_Oper (N_Paciente, Cod_Operação)

5.5 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.

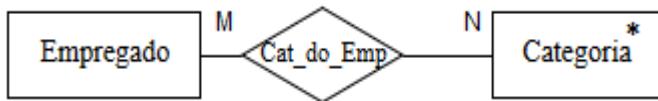
2. A entidade *Categoria* é obrigatória na associação?

Se a lista de categorias inclui valores que nenhum empregado possui, deverá manter-se.

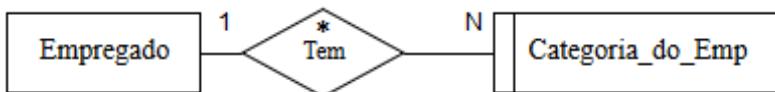
Caso contrário, se as únicas categorias que interessam são as dos empregados existentes pode eliminar-se a tabela.

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.

5.6 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 excede 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.

5.7 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 selecionar 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.

5.8 Exemplo – Biblioteca

Uma biblioteca guarda registo sobre os livros existentes e sobre os empréstimos aos seus sócios.

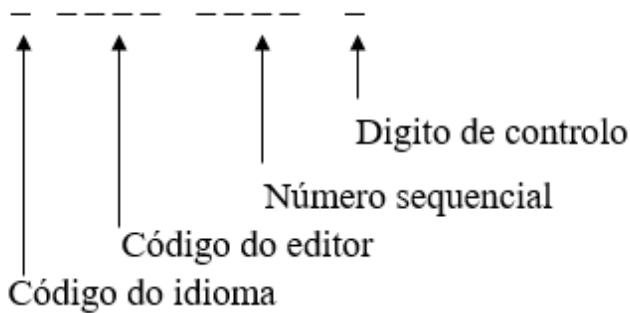
Cada sócio é identificado por um *número de sócio* e cada exemplar de livro por *número de exemplar* (pode existir mais do que um exemplar do mesmo livro).

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 1^a lista de transações

1. Inserir detalhes de novos sócios
2. Inserir detalhes de novas aquisições
3. Fazer um empréstimo
4. Registar 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 1^a 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 $ISBNX = \{cod_editor, n_série\}$

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).

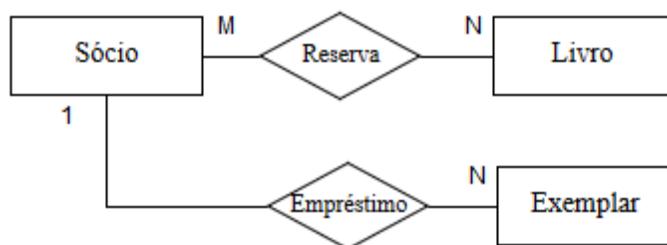
Um sócio pode fazer várias reservas e o mesmo livro pode ser reservado por vários sócios.

Uma **1^a 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.

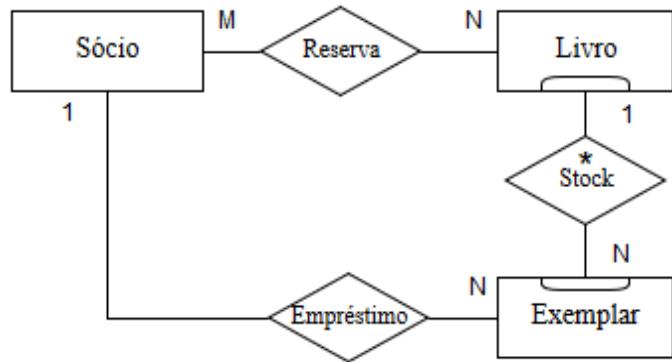
O diagrama pode ser alterado por introdução da entidade Livro e da associação reserva:



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)

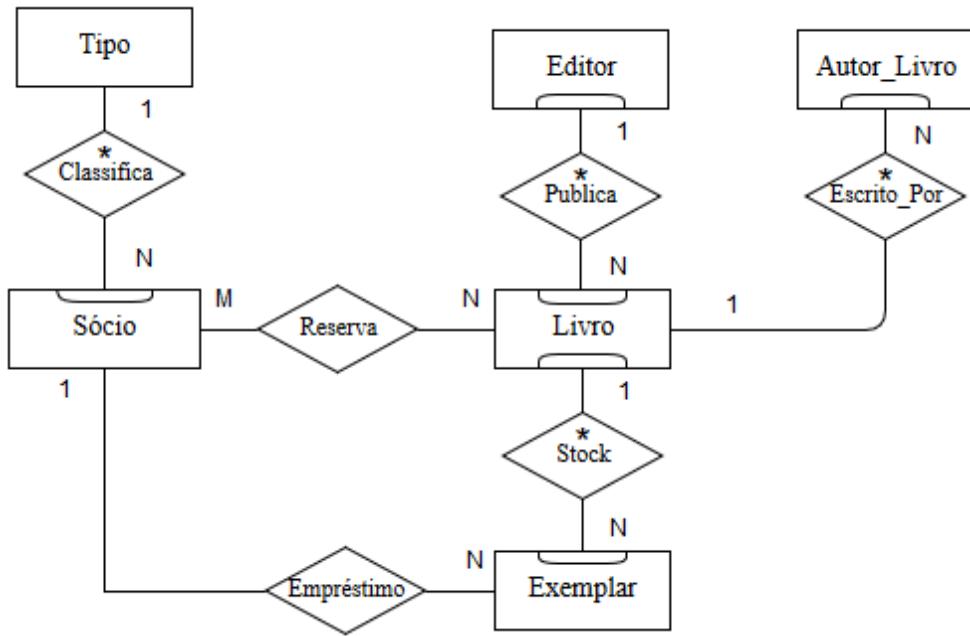
Reserva (n_sócio, ISBNX, data_reserva)

- 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.

6 Normalização avançada

6.1 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 $\Rightarrow\!\!\!\rightarrow$ cor)

e que item multidetermina departamento (item $\Rightarrow\!\!\!\rightarrow$ departamento)

Seja a relação, $R(X, Y, Z)$ com $m + n + r$ atributos, onde

$$\begin{aligned}X &= \{X_1, X_2, \dots, X_m\}, \\Y &= \{Y_1, Y_2, \dots, Y_n\}, \\Z &= \{Z_1, Z_2, \dots, Z_r\},\end{aligned}$$

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

$$cor_{c1} = \{castanho\}$$

$$cor_{tl} = \{preto\}$$

$$cor_{s1} = \{castanho\}$$

$$cor_{d2} = \{verde\}$$

$$cor_{s2} = \{castanho\}$$

$$cor_{b2} = \{vermelho, azul, amarelo\}$$

$$cor_{b3} = \{vermelho, azul, amarelo\}$$

6.1.1 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*

$$cor_{s1} = cor_{s2} = \{castanho\}$$

$$cor_{b2} = cor_{b3} = \{vermelho, azul, amarelo\}$$

$$item \twoheadrightarrow cor$$

$Y_{xz} = Y_{x'z'}$, verifica-se sse sempre que (x, y, z) e (x', y', z') são tuplos de $R(XYZ)$ também o são (x, y', z) e (x, y, z') .

6.1.2 Definição 2 (DM)

A dependência multivalor, $X \twoheadrightarrow Y$ existe em $R(XYZ)$ sse sempre que (x, y, z) e (x', y', z') são tuplos de $R(XYZ)$ também o são (x, y', z) e (x, y, z') .

Por exemplo,

$(b, 2, vermelho)$ e $(b, 3, azul)$ são tuplos de *Inventário* assim como
 $(b, 2, azul)$ e $(b, 3, vermelho)$

X multidetermina Y se um valor de Y identifica um conjunto de valores de Y independentemente de outros atributos da relação.

6.1.3 DFs e DM

Se X e Y são conjuntos disjuntos de atributos da relação $R(XYZ)$ tal que existe $X \rightarrow Y$ então também $X \twoheadrightarrow Y$.

6.1.4 DM Complementares

Se a DM $X \twoheadrightarrow Y$ está definida na relação $R(XYZ)$ então também existe $X \twoheadrightarrow Z$.

Na relação *Inventário*, se $Item \rightarrow\!\!\! \rightarrow Cor$ então $Item \rightarrow\!\!\! \rightarrow Departamento$ (e vice-versa)

De facto,

$$\begin{aligned} Departamento_{b, vermelho} &= Departamento_{b, amarelo} \\ &= Departamento_{b, azul} = \{ 2, 3 \} \end{aligned}$$

6.1.5 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?

6.1.6 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 $x y z \in R(X, Y, Z) \subseteq \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$. Então existem y' e z' tais que $x y z'$ e $x y' z \in R(X, Y, Z)$.

Mas $X \twoheadrightarrow Y$, donde, por definição de DM, $x y z$ e $x y' z' \in R(X, Y, Z)$.

Portanto, $x y z \in R(X, Y, Z)$.

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 $x y z$ e $x y' z'$ quaisquer tuplos de $R(X, Y, Z)$.

Então $x y$ e $x y' \in \pi_{X, Y}(R)$, com $x z$ e $x z' \in \pi_{X, Z}(R)$.

Por junção de $x y$ e $x y'$ com $x z$ e $x z'$ obtemos $x y z$, $x y z'$, $x y' z$ e $x y' z'$.

Porque assumimos que $R(X, Y, Z) = \pi_{X, Y}(R) \bowtie \pi_{X, Z}(R)$

então $x y' z \in R(X, Y, Z)$ e $x y' z' \in R(X, Y, Z)$.

Donde, pela definição de dependência multivalor, $X \twoheadrightarrow Y$ em $R(X, Y, Z)$.

6.1.7 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.

6.2 Dependências de Junção (DJ)

6.2.1 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:

$R (Fornecedor, Peça, Projeto)$

Fornecedor	Peça	Projeto
F1	P1	J2
F1	P2	J1
F2	P1	J1
F1	P1	J1

existe a dependência de junção,

$*\{\{Fornecedor, Peça\}, \{Peça, Projeto\}, \{Projeto, Fornecedor\}\}$

$\pi_{\text{Fornecedor, Peça}}(R)$	$\pi_{\text{Peça, Projeto}}(R)$	$\pi_{\text{Projeto, Fornecedor}}(R)$		
Fornecedor	Peça	Projeto	Projeto	Fornecedor
F1	P1	P1	J2	J2
F1	P2	P2	J1	J1
F2	P1	P1	J1	J1

$\pi_{\text{Fornecedor, Peça}}(R) \bowtie \pi_{\text{Peça, Projeto}}(R)$:

Fornecedor	Peça	Projeto
F1	P1	J2
F1	P1	J1
F1	P2	J1
F2	P1	J2
F2	P1	J1

$\pi_{\text{Fornecedor, Peça}}(R) \bowtie \pi_{\text{Peça, Projeto}}(R) \bowtie \pi_{\text{Projeto, Fornecedor}}(R)$:

Fornecedor	Peça	Projeto
F1	P1	J2
F1	P2	J1
F2	P1	J1
F1	P1	J1

6.2.2 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 .

6.2.3 Dependências de Junção e Dependências baseadas em Chaves

Diz-se que a DJ $*\{X_1, X_2, \dots, X_m\}$ 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 $*\{AB, AD, BC\}$ é baseada em chaves!

6.3 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 $E1$ (Peça, Departamento) e $E2$ (Peça, Cor).

A DM $Peça \twoheadrightarrow Departamento$ garante que $Inventário = E1 \bowtie E2$.

6.3.1 4^a Forma Normal (4FN)

Uma relação $R (X, Y, Z)$, com X, Y e Z conjuntos de atributos disjuntos, está na 4FN sse $\forall X \twoheadrightarrow Y$, não trivial: X é chave candidata de R .

6.3.2 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 .

$$X \rightarrow Y \implies X \rightarrow YI$$

$X \rightarrow YI \wedge X \cap YI = \emptyset \implies X \twoheadrightarrow YI$ 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.

$X \cup YI$ não é o conjunto de todos os atributos porque A não é de X nem de YI .

- (. se A pertencesse a X então X determinava A , o que contradiz a nossa hipótese;
- . se A pertencesse a YI , como $X \rightarrow Y$ e $YI \subseteq Y$ mais uma vez significava que X determinava A)

Portanto concluímos que R tem uma dependência multivalor não trivial $X \twoheadrightarrow YI$ e um atributo A tal que $X \not\rightarrow A$ (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:

$R (\underline{Curso}, \underline{Professor}, \underline{Livro})$

$\underline{Curso} \twoheadrightarrow \underline{Livro}$

$\underline{Curso} \twoheadrightarrow \underline{Professor}$

Chave: $\underline{Curso}, \underline{Professor}, \underline{Livro}$

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. \underline{Curso} não é chave candidata, mas a DM é trivial.

6.4 Quinta Forma Normal

(ou forma normal de projeção/junção – FNPJ)

6.4.1 Definição:

Uma relação R está na 5^aFN 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

$$\pi_{Agente, Companhia}(R)$$

$$\pi_{Agente, Produto}(R)$$

$$\pi_{Companhia, Produto}(R)$$

6.4.2 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 decomponemos 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, 2^a Edição. FCA.
- Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems, 3th Edition. McGraw-Hill.

Apêndice – Trabalhos Práticos (anos anteriores)

Alguns trabalhos práticos desenvolvidos em anos anteriores.

Ano lectivo 2013/2014

Embora a componente prática da disciplina corresponda a sete valores na nota final, individualmente cada aluno pode ter uma classificação diferente.

Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se avaliar a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida em grupo. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelo docente.

O sistema de gestão de base de dados a usar para a realização do trabalho é o *SQL Server 2000*.

A aplicação pode ser desenvolvida em java (*JBuilder, NetBeans, Delphi ou C++*).

1. Reserva/requisição de equipamento do DI-UBI (Descrição da organização)

No Departamento de Informática da UBI (DI-UBI) existem diversos equipamentos (portáteis, projectores de vídeo, etc.) que são partilhados pelos potenciais utilizadores. A utilização destes recursos é regulada por uma política de reservas/requisições.

Utilizadores e prioridade:

Os utilizadores encontram-se agrupados por *tipo de utilizador*. Cada tipo de utilizador é identificado por um prefixo constituído por duas letras. Alguns prefixos já estão atribuídos: PR – professor, RS – investigador, BS – estudante de licenciatura, MS – estudante de mestrado, DS – estudante de doutoramento, SF – pessoal de apoio e XT – externo.

Os utilizadores são identificados por IDs (10 chars). Os primeiros três caracteres do ID são ‘XX_’, onde XX é o prefixo (por exemplo, “RS_Special” é o ID de um investigador). Para cada utilizador é armazenado o seu número de telefone e/ou o seu endereço de correio electrónico.

Classes de prioridades e prioridade corrente

Cada tipo de utilizador tem associado uma *classe de prioridade*. São admitidas as seguintes classes de prioridade: máxima, acima da média, média, abaixo da média e mínima.

Os professores têm classe de prioridade “acima da média”, enquanto os outros utilizadores têm classe de prioridade “média”. O Presidente do Departamento é um utilizador especial, donde a sua prioridade é sempre “máxima”.

Cada utilizador tem atribuído uma *prioridade corrente*, inicialmente igual ao valor da classe de prioridade. A prioridade corrente pode variar (subir ou descer) com o passar do tempo, ou seja, a prioridade corrente é dinâmica.

Reservas:

O utilizador que faz (coloca) a *reserva* é o responsável pela mesma. Quando uma reserva é colocada, é-lhe atribuído um selo temporal (i.e., é registado o seu

timestamp - dia e hora) e é gerado um *ID* (8 *chars*) para a identificar. O *ID* tem o seguinte formato: yyyySSSS, onde yyyy é o ano corrente e SSSS é um número sequencial para o ano yyyy. Por exemplo, 20130001 é o *ID* da primeira reserva colocada no ano 2013.

Ao fazer a reserva, o utilizador indica o dia e hora em que pretende os recursos e durante quanto tempo os vai utilizar, i.e., indica o *período de uso*. Para cada recurso pretendido, indica também se este é ou não *imprescindível*, ou seja, se o recurso é fundamental para a reserva fazer sentido. Por exemplo, um projector de vídeo pode ser indispensável, mas uma extensão eléctrica pode não o ser.

Estado:

Uma reserva pode estar no estado *active* (activa), *satisfied* (satisfeita), *canceled* (cancelada), *waiting* (à espera), ou *forgotten* (esquecida).

Está no estado *active* quando os recursos estão disponíveis para o período pretendido. Está no estado *waiting* quando pelo menos um dos equipamentos pretendidos não está disponível.

Mudanças de estado:

A partir do estado *active*, uma reserva pode transitar para o estado *satisfied*, *canceled*, *waiting*, ou *forgotten*. A partir do estado *waiting* pode transitar para o estado *active*, *satisfied*, *canceled* ou *forgotten*. Os estados *satisfied*, *canceled* e *forgotten* são estados finais (i.e., quando uma requisição entra num destes estados já não pode transitar para outro estado).

Quando uma reserva é colocada fica no estado *active* ou *waiting*. Transita dos estados *active* ou *waiting* para o estado *satisfied* se os equipamentos reservados forem levantados antes do fim do “período de uso”. Transita para o estado *canceled* se a reserva for cancelada. Transita para o estado *forgotten* quando o “período de uso” se esgota sem que a reserva tenha sido *satisfied* ou *canceled*.

A transição do estado *active* para *satisfied* corresponde a levantar todos os equipamentos reservados. A transição do estado *waiting* para *satisfied* corresponde a levantar os equipamentos reservados e que se encontrem disponíveis.

Nota: tal como a colocação de novas reservas pode levar a mudanças no estado de outras reservas, o cancelamento de uma reserva também pode provocar alterações no estado de outras reservas (aqueles que estão no estado de *waiting*).

Preempção:

A atribuição de recursos segue uma regra de preempção baseada na prioridade corrente dos utilizadores.

A colocação duma nova reserva (por um utilizador prioritário!) pode fazer com que outra transite do estado *active* para o estado *waiting*, e nesse caso diz-se que ocorreu uma preempção. De um modo geral, uma preempção só pode ocorrer até 48 horas antes do início do “período de reserva”. A partir daí só o Presidente do Departamento pode provocar uma preempção (o Presidente do Departamento pode fazer reservas/requisições em qualquer altura).

Requisição:

Quando uma reserva passa para o estado *satisfied* é gerada uma requisição para os equipamentos levantados.

Equipamentos levantados sem reserva também dão origem a uma requisição.

Uma requisição está no estado *active* enquanto os equipamentos levantados não forem devolvidos. Nessa altura passa para o estado *closed*.

Variação da prioridade dos utilizadores:

Uma reserva no estado *active* ou *waiting* (com equipamento essencial atribuído) não levantada até ao final do período de uso implica uma descida na prioridade corrente do utilizador.

Cancelamento de uma reserva: até 2 horas antes do início do período de uso não tem qualquer penalização na prioridade; até ao início do período com uma “falta” na prioridade. O cancelamento dentro do período de uso é penalizado com uma falta por cada hora (com um máximo de 3 faltas).

Entrega dos recursos levantados: até 15 minutos após o período de uso não tem qualquer penalização; por cada hora de atraso o utilizador será penalizado com uma falta na prioridade. O utilizador pode ir entregando os recursos levantados, não sendo, portanto, necessário entregá-los todos de uma vez.

Cinco faltas na prioridade provocam uma descida no nível da prioridade corrente.

A prioridade corrente dum utilizador sobe um nível após tratar correctamente duas reservas/requisições consecutivas. O valor da prioridade corrente de um utilizador nunca pode ultrapassar o valor da sua classe de prioridade.

Nota final:

A todo o momento é importante conhecer o estado de cada equipamento: disponível, em uso (e por quem) ou reservado (e por quem).

2. Tarefas a realizar

2.1 Elaborar o modelo de dados para o sistema de reservas/requisições de equipamento do DI-UBI.

Notas:

1. Assumir para as situações não especificadas as soluções que pareçam mais plausíveis. Indicar explicitamente as escolhas efectuadas.
2. O modelo deve estar normalizado.

2. Produzir *scripts* para:

- Criar a base de dados;
- Criar tabelas e restrições tendo em conta o modelo de dados desenvolvido;
Não se esqueça de estabelecer a chave primária (e as chaves estrangeiras, se existirem).
Criar as restrições *UNIQUE/Check/Not Null* que sejam necessárias. Tenha em atenção a gama de valores que os atributos podem assumir.
- Criar procedimentos armazenados e funções definidas pelo utilizador (UDFs);
Crie, entre outros, o *Stored Procedure Reserve2Requisition* que recebe como parâmetro de entrada o ID de uma reserva e que cria a correspondente requisição.

Crie, entre outras, a função *MakeID* que recebe como parâmetros de entrada a data e um número e devolve um *string* (8 chars) com o ID de uma reserva (requisição).

- Criar *triggers* e garantir o cumprimento das “regras do negócio”;

Crie, por exemplo, o *trigger* para associar ao evento UPDATE da tabela das reservas.

Na actualização do estado da reserva para *satisfied*, o *trigger* deve executar o procedimento *Reserve2Requisition*. ~~Se a transição de estado for para um estado final deve eliminar as “linhas” da reserva.~~

Nota: repare que vai precisar de criar vários *triggers* para garantir o cumprimento das “regras do negócio”. ~~Por exemplo, tem que garantir que o prefixo que entra na composição do ID é um prefixo válido.~~

- Criar *views*;

Crie, entre outras, a *view ResourceState* para nos apresentar os dados no seguinte formato:

ResID	ResDesc	State	ID	User
1	Asus LC3	Available		
12	Sony DCR405	Reserved	20130048	BS_Dragon
24	Toshiba	InUse	20130097	RS_Special
...

ID → ID de uma reserva ou ID de uma requisição!

- Inserir alguns dados iniciais (dados de arranque).

Inserir dados sobre utilizadores, tipos de utilizador e níveis de prioridade, recursos, estados, etc..

Nota: a aplicação a desenvolver não deve inserir dados nessas tabelas!

3. Construir uma aplicação que permita:

- Mostrar os recursos e o seu estado (e eventuais utilizadores).

- Gerir as reservas: registar uma reserva; alterar o estado duma reserva.

Nota: não esquecer que algumas mudanças de estado são automáticas, como por exemplo, a passagem para o estado *waiting*.

- Gerir as requisições: registar uma requisição; aceitar a devolução dos equipamentos.

Nota: não esquecer que estas acções podem provocar efeitos secundários...

4. Elaborar um relatório com a descrição permonorizada do trabalho realizado

3. Elementos a entregar, datas e cotação do trabalho

...

4. Estrutura do Relatório

O relatório deve conter, pelo menos, os seguintes capítulos (a adaptar de acordo com a versão escolhida para desenvolver a aplicação):

Prefácio

Incluir a identificação dos elementos do grupo.

1. Introdução

Apresentação do trabalho desenvolvido e introdução genérica sobre as ferramentas utilizadas.

2. Modelo de dados e *scripts*:

Fazer uma breve introdução à modelação de dados e à construção do modelo conceptual.
Apresentar e justificar o modelo de dados desenvolvido.

Deve incluir: uma descrição da organização (tal como entendida pelo grupo); as opções tomadas para as situações não especificadas no enunciado; indicar as “regras de negócio” da organização.

3. Descrição da aplicação

3.1 Decomposição do trabalho

Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.

Cada elemento do grupo deve incluir uma subsecção com uma descrição muito precisa das suas tarefas. Pode incluir pseudo-código se tal for necessário. Em anexo, devidamente identificado com o número de aluno, podem ser incluídos mais elementos que o estudante considere relevantes (por exemplo, código SQL).

3.2 Configuração da ligação à base de dados

Mostrar como se configurou a ligação à base de dados. Incluir ecrãs ilustrativos -

3.3 Capacidades da aplicação

3.3.1 Ligação à base de dados

Código ilustrando a forma como foi estabelecida a ligação à base de dados

3.3.2 Acessos à base de dados

Documentar devidamente a forma como foi efectuado o acesso à base de dados. Incluir secções de código ilustrando o acesso à base de dados (exemplos para operações de consulta, inserção, eliminação e actualização).

3.3.3 Listagens

Resultados mais significativos da aplicação.

3.4 Funcionalidade

(Nesta parte apresentar uma breve descrição da funcionalidade da aplicação – Tarefa3)

3.4.1 Mostrar a inserção de uma reserva (e respectivos recursos)

- Do ponto de vista do utilizador (incluir imagens representativas – *screenshots*)
- Do ponto de vista do programador (incluir código e SQL correspondente)

4.2 Explicar como se muda o estado de uma reserva

- Do ponto de vista do utilizador (incluir imagens representativas – *screenshots*)
- Do ponto de vista do programador (desde a aplicação até à base de dados). Incluir código e SQL correspondente. Exemplificar para cada transição de estado o que está a acontecer na base de dados.

4.3 Mostrar a inserção de uma requisição

- Do ponto de vista do utilizador (incluir imagens representativas – *screenshots*)
- Do ponto de vista do programador (desde a aplicação até à base de dados). Incluir código e SQL correspondente.

4.4 Explicar como se procede à devolução de equipamento

- Do ponto de vista do utilizador (incluir imagens representativas – *screenshots*)
- Do ponto de vista do programador (desde a aplicação até à base de dados). Incluir código e SQL correspondente.

4.5 Explicar como se pode conhecer o estado dos recursos quanto à sua disponibilidade para um dado período

- Do ponto de vista do utilizador (incluir imagens representativas – *screenshots*)
- Do ponto de vista do programador (desde a aplicação até à base de dados). Incluir código e SQL correspondente.

4.6 Outras funcionalidades desenvolvidas

4. Conclusões

Indicar o que foi conseguido.

Indicar o que não foi conseguido. Indicar a(s) razão(ões).

Incluir uma reflexão crítica sobre a disciplina (pontos a manter, a alterar e a eliminar).

Apêndices

A- Scripts (criar bd, criar tabelas e restrições, triggers; procedimentos armazenados; dados iniciais)

B- Para cada aluno: listagem com o código (SQL e Java – ou outro) que desenvolveu. Incluir screenshots dos ecrãs desenvolvidos.

Ano lectivo 2014/2015

1. Zurrapa – Drinks & Coffee (Descrição da organização)

A empresa *Zurrapa-Drinks & Coffee, Lda.* dedica-se exclusivamente ao comércio de bebidas e café em estabelecimentos de grandes dimensões (universidades, hospitalares, etc.). Suponha que no próximo ano, a *Zurrapa* vai gerir/explorar os bares da UBI.

A Zurrapa definiu que todos os produtos são encaminhados a partir de um só ponto de distribuição, designado por Cais/Armazém, para os diversos bares. Portanto, os fornecedores entregam os produtos (bebidas e café) no Cais e aí são armazenados. Mais tarde esses produtos são encaminhados para os bares que deles necessitam.

A empresa tem vários empregados. Cada empregado está atribuído a um bar, que corresponde ao seu local de trabalho pré-definido, e pode assumir um de dois papéis: empregado de balcão ou empregado de mesa. De dia para dia, os empregados podem ser deslocados do seu local de trabalho pré-definido para outro bar ou para o Cais/Armazém. Cada bar tem um responsável, que pode variar ao longo do ano.

Os empregados de mesa estão equipados com *Tablets* (computador de pequenas dimensões) onde registam, para cada mesa, os produtos solicitados. O empregado de mesa que inicia o serviço numa mesa é responsável pelo mesmo, ou seja, o empregado que “abre” a mesa é responsável por ela.

O empregado de balcão consulta os pedidos “em aberto”, e, em função deles, prepara os produtos solicitados e entrega-os ao empregado de mesa, ficando o pedido marcado como satisfeito. Quando o serviço é pago, o pedido é fechado.

A receção de mercadorias no Cais/Armazém, o seu armazenamento e posterior distribuição pelos bares é feita por um empregado, designado por empregado do Cais. A Zurrapa tem por prática selecionar, de modo aleatório, o empregado do Cais somente entre os seus empregados de mesa (ou seja, os empregados de balcão não prestam serviço no Cais). O escalonamento do empregado do Cais realiza-se à sexta-feira e é válido para toda a semana seguinte.

No Cais/Armazém então armazenados, sobretudo, produtos agregados (caixas e/ou embalagens) enquanto nos bares estão produtos individualizados. Por exemplo, no armazém encontram-se grades de cerveja (contendo 24 cervejas), embalagens com 6 garrafas 1.5L de água, sacos com 1 Kg de café (correspondente a 60 cafés individuais), etc.. Num bar podem encontrar-se 28 cervejas, 7 águas de 1.5L e “17” cafés. O empregado do Armazém é o responsável por verificar o *stock* dos bares e fazer a reposição dos produtos em falta.

A gerência pretende saber, a cada instante, o valor total (€) dos produtos que tem no Armazém e o valor total (€) dos produtos que tem em cada bar – estes valores reportam-se a preços de custo. A gerência pretende saber, para cada dia, quanto foi gasto (a preço de custo) e quanto foi recebido em cada bar.

2. Tarefas a realizar

2.1 Elaborar o modelo de dados para o sistema descrito (tendo também em atenção as transações que é necessário satisfazer – ver ponto 3).

Notas:

1. Assumir para as situações não especificadas as soluções que pareçam mais plausíveis. Indicar explicitamente as escolhas efectuadas.
2. O modelo deve estar normalizado.

2.2 Produzir *scripts* para:

- Criar a base de dados;
- Criar tabelas e restrições tendo em conta o modelo de dados desenvolvido;
 - Não se esqueça de estabelecer a chave primária (e as chaves estrangeiras, se existirem).
 - Criar as restrições *UNIQUE/Check/Not Null* que sejam necessárias. Tenha em atenção a gama de valores que os atributos podem assumir.
- Inserir alguns dados iniciais (dados de arranque).

2.3 Construir aplicações para:

- a) O empregado de mesa “tratar” os pedidos dos clientes. Em caso dos produtos solicitados não existirem em quantidade suficiente no bar, a aplicação deve informar o utilizador (empregado de mesa) se esses produtos existem em armazém e nesse caso quanto tempo demora a ser reposto o stock. Note-se que o empregado de mesa é o responsável por abrir/fechar as mesas.
- b) A aplicação para o empregado de balcão “tratar” os pedidos. Esta aplicação é responsável por actualizar o stock do bar.
- c) A aplicação do empregado do Cais. Dentro das funcionalidades da aplicação estão: carregar os stocks do armazém; consultar/carregar os stocks dos bares; produzir os dados para a Gerência.

2.4 Elaborar um relatório com a descrição pormenorizada do trabalho realizado

3. Elementos a entregar, datas e cotação do trabalho

...

4. Estrutura do Relatório

O relatório deve conter, pelo menos, os seguintes capítulos:

Prefácio

Incluir a identificação dos elementos do grupo.

1. Introdução

Apresentação do trabalho desenvolvido e introdução genérica sobre as ferramentas utilizadas.

2. Modelo de dados e *scripts*:

Fazer uma breve introdução à modelação de dados e à construção do modelo conceptual.
Apresentar e justificar o modelo de dados desenvolvido.

Deve incluir: uma descrição da organização (tal como entendida pelo grupo); as opções tomadas para as situações não especificadas no enunciado; indicar as “regras de negócio” da organização.

3. Aplicação

3.1 Decomposição e distribuição de tarefas

Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.

3.1.xx Descrição precisa das tarefas (cada elemento do grupo faz a sua)

Cada elemento do grupo deve incluir uma subsecção com uma descrição muito precisa das suas tarefas. Pode incluir pseudo-código se tal for necessário. Em anexo, devidamente identificado com o número de aluno, podem ser incluídos mais elementos que o estudante considere relevantes (por exemplo, código SQL).

3.2 Acesso à base de dados

Documentar, devidamente, a forma como foi efectuado o acesso à base de dados. Incluir secções de código ilustrando o acesso à base de dados (exemplos para operações de consulta, inserção, eliminação e actualização).

3.3 Funcionalidade

3.3.1 Descrição geral

Descrição da funcionalidade global, incluindo uma representação esquemática de como funciona a solução (BD; atendimento nas mesas, atendimento no balcão e interação cais-bares e cais-armazenamento).

3.3.1 Receção de pedidos – empregado de mesa

Descrição da funcionalidade, focando os pontos de vista do utilizador (incluir imagens representativas – *screenshots*) e do ponto de vista do programador (incluir excertos de código e SQL).

3.3.2 Balcão bar – Satisfação dos pedidos

Descrição da funcionalidade, focando os pontos de vista do utilizador (empregado de balcão) e do ponto de vista do programador (incluir excertos de código e SQL).

3.3.2 Interação Bar-Armazém

Descrição da funcionalidade, focando os pontos de vista do utilizador e do ponto de vista do programador (incluir excertos de código e SQL). Ilustrar, sobretudo, o modo como o *stock* do armazém é incrementado (recepção de produtos) e como o stock dos bares é reforçado.

3.3.3 Outras funcionalidades desenvolvidas

4. Conclusões

Indicar o que foi conseguido.

Indicar o que não foi conseguido. Indicar a(s) razão(ões).

5. Epílogo

Incluir uma reflexão crítica sobre a disciplina (pontos a manter, a alterar e a eliminar).

Apêndice

Scripts (criar bd, criar tabelas e restrições, dados iniciais)

5. Notas finais

- 1) Embora a componente prática da disciplina corresponda a sete valores na nota final, individualmente cada aluno pode ter uma classificação diferente.

Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se aferir a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida dentro do grupo de trabalho. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelos docentes.

- 2) O SGBD a usar é o *SQL Server 2012 Express Edition* (ou outro, desde que aceite pelos docentes).

As aplicações devem ser desenvolvidas em java - *NetBeans* (ou outra, desde que aceite pelos docentes).

Ano lectivo 2015/2016

1. Enquadramento “Go ahead, make my day”

A empresa *MakeMyDay-Drinks & Food, Lda.* encontra-se instalada em diversos espaços comerciais, onde põe à disposição dos clientes serviços de restauração e bebidas. Em cada filial é possível encontrar as seguintes áreas funcionais: uma Cozinha, duas salas de refeições (Sala Pequena e Sala Grande – também conhecida como Sala de Eventos), um Bar, uma Esplanada e um Armazém. Cada área funcional tem atribuída uma equipa de colaboradores, chefiada por um responsável pertencente à equipa.

As áreas funcionais (exceto Cozinha e Armazém) têm várias mesas. Cada mesa tem vários lugares sentados (2, 3, 4, ...). As mesas das salas estão dispostas em matriz (por exemplo, a mesa 23 pertence à linha 2 coluna 3). No Bar é possível fazer refeições no balcão e nas mesas. O balcão tem um dado número de lugares sentados (bancos).

No Bar, os pedidos dos clientes são recebidos pelos empregados de balcão. Nas salas, os clientes são encaminhados para uma mesa apropriada, tendo em conta o número de pessoas, sendo os pedidos recebidos por um colaborador. Na Esplanada, a escolha de mesa é deixada ao critério do cliente, sendo os pedidos recebidos por qualquer colaborador.

Sandes e pratos (da ementa) são sempre preparados na Cozinha. O serviço de bebidas é sempre executado a partir do Bar. A reposição do *stock* do Bar é feita a partir do Armazém.

Na Cozinha e no Bar existem impressoras para imprimir “os pedidos” dos clientes. Como um pedido pode envolver várias bebidas e “comidas” (sandes, entradas, pratos confeccionados, etc.), um mesmo pedido de cliente pode originar a impressão de vários *talões* (Bar e Cozinha). O serviço de refeições deve tentar servir os pratos em simultâneo. Caso seja previsível algum atraso na preparação dum prato, o cliente deve ser avisado e deve ser incluída uma nota (no talão da cozinha) a indicar a preferência do cliente (servido em conjunto ou servido logo que possível).

Como é usual, os clientes podem fazer “novos” pedidos à medida que a refeição avança. Deve também considerar-se o caso de novos pedidos após o cliente ter solicitado “a conta”.

Gestão dos pedidos dos clientes: 1) o pedido é aceite por um colaborador; 2) os itens do pedido são preparados (pelos respetivos responsáveis das áreas funcionais envolvidas); 3) os itens são servidos ao cliente por um colaborador; 4) a “conta” pode ser solicitada a qualquer colaborador; 5) o talão de conferência de consumos é emitido pelo responsável do Bar (onde também está instalado o sistema POS – *Point Of Sale*) e devolvido pelo responsável da mesa ao cliente; 6) o dinheiro do pagamento é levado por um colaborador junto do POS e é preparada a fatura e o troco, caso haja lugar ao mesmo; 7) o responsável de mesa devolve a fatura (e o troco) ao cliente. O colaborador que aceita o primeiro pedido para uma mesa é o responsável pela mesa (ou seja, é aquele que faz a interface com o cliente e também o que recebe a gorjeta!).

As encomendas de produtos ao exterior são tratadas diretamente pela Gerência e, portanto, não são do domínio público.

A gerência pretende saber, para cada categoria de produto (café e chá, refrigerante, cerveja, vinho, sandes, pratos, etc.), o valor total (€) recebido em cada unidade funcional.

Nota: para a gerência é muito importante que os clientes sejam servidos pela ordem em que os pedidos são submetidos.

2. Tarefas a realizar

2.1 Elaborar o modelo de dados para o sistema descrito.

Notas:

1. Assumir para as situações não especificadas as soluções que pareçam mais plausíveis. Indicar explicitamente as opções tomadas.
2. O modelo deve estar normalizado (3FN).

2.2 Base de dados

Implementar a base de dados em SQL Server (ou outro, desde que aceite pelo docente). **Producir scripts** para:

- Criar a base de dados;
- Criar tabelas e restrições tendo em conta o modelo de dados desenvolvido;
- Inserir alguns dados iniciais (dados de arranque).

2.3 Aplicações

A. Desenvolver a aplicação “Mesa” para “tratar” dos pedidos dos clientes.

A aplicação cliente/servidor deve permitir:

- a) Abrir mesa.
- b) Inserir produtos.
- c) Pedir conta.
- d) Fechar mesa.
- e) Marcar mesa como pronta (i.e., limpa e pronta para receber clientes).

B. Desenvolver a aplicação “Bar”.

A aplicação cliente/servidor deve permitir, pelo menos:

- a) Mostrar os talões do bar (com o número de ordem e as bebidas envolvidas);
- b) Marcar o pedido como preparado (i.e., as bebidas estão prontas).
- c) Dar baixa no stock correspondente.

C. Desenvolver a aplicação “POS”.

A aplicação cliente/servidor deve permitir, pelo menos:

- a) Listas as contas por pagar;
- b) Marcar uma conta como “Paga”.

D. Desenvolver a aplicação “Cozinha”.

A aplicação cliente/servidor deve permitir, pelo menos:

- a) Mostrar os talões por preparar (com o número de ordem e os pratos);
- b) Marcar os pratos/sandes como preparados (i.e., indica que já estão prontos a servir).

Nota: A numeração dos talões deve ser sequencial.

2.4 Relatório

Elaborar um relatório com a descrição permanorizada do trabalho realizado.

3. Elementos a entregar, datas e cotação do trabalho

3.1 Elementos a entregar

- ~~Relatório em papel~~
- Relatório em formato digital (.pdf e .doc)
- Scripts em formato texto (.txt ou .sql). Os scripts devem ser escritos manualmente, tal como usados nas aulas práticas. A entrega dos scripts obtidos diretamente a partir do SGBD será penalizada.
- *Source code* das aplicações ~~em formato digital~~
- Scripts para compilação do *source code* (em alternativa configuração de projeto em *Eclipse* ou *Netbeans* para replicar o processo de compilação)
- ... **(em desenvolvimento)**

3.2 Datas importantes

Relatório em papel: 20 de Maio de 2016 (hora de expediente).

Modelo de dados, Scripts, Relatório e aplicação (via *moodle* ???): 22 de Maio de 2016.

Discussão: aulas práticas da semana 23-27 de Maio.

...

4. Estrutura do Relatório

O relatório deve conter, pelo menos, os seguintes capítulos:

Capa

Incluir a identificação dos elementos do grupo.

1. Introdução

Introdução ao trabalho desenvolvido e introdução genérica sobre as ferramentas utilizadas.

2. Modelo de dados e *scripts*:

Fazer uma breve introdução à modelação de dados e à construção do modelo conceptual. Apresentar e justificar o modelo de dados desenvolvido.

Deve incluir: uma descrição da organização (tal como entendida pelo grupo); as opções tomadas para as situações não especificadas no enunciado; indicar as “regras de negócio” da organização.

Deve incluir a imagem (foto) do modelo de dados solicitada (e assinada) pelo docente na semana 2-6 de maio.

3. Aplicação

3.1 Decomposição e distribuição de tarefas

Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.

3.1.xx Descrição precisa das tarefas (cada elemento do grupo faz a sua)

Cada elemento do grupo deve incluir uma subsecção com uma descrição muito precisa das suas tarefas. Pode incluir pseudo-código se tal for necessário. Em anexo, devidamente identificado com o número de aluno, podem ser incluídos mais elementos que o aluno considere relevante para melhor caracterizar o seu trabalho (por exemplo, código SQL).

3.2 Acesso à base de dados

Documentar, devidamente, a forma como foi efectuado o acesso à base de dados. Incluir secções de código ilustrando o acesso à base de dados (exemplos para operações de consulta, inserção, eliminação e actualização).

3.3 Funcionalidade

3.3.1 Descrição geral

Descrição da funcionalidade global, incluindo uma representação esquemática de como funciona a solução (BD; atendimento nas mesas, atendimento no balcão, atendimento na esplanada, interação pedidos/bar/cozinha, pagamentos, etc.).

3.3.1 Receção de pedidos – Aplicação Mesa

Descrição da funcionalidade, focando os pontos de vista do utilizador (incluir imagens representativas – *screenshots*) e do ponto de vista do programador (incluir excertos de código e SQL).

3.3.2 Bar – Satisfação dos pedidos

Descrição da funcionalidade, focando os pontos de vista do utilizador (empregado de balcão) e do ponto de vista do programador (incluir excertos de código e SQL).

3.3.3 POS - Pagamentos

Descrição da funcionalidade, focando os pontos de vista do utilizador e do ponto de vista do programador (incluir excertos de código e SQL).

3.3.4 Cozinha – Satisfação dos pedidos

Descrição da funcionalidade, focando os pontos de vista do utilizador e do ponto de vista do programador (incluir excertos de código e SQL).

3.3.4 Outras funcionalidades desenvolvidas

4. Conclusões

Indicar o que foi conseguido.

Indicar o que não foi conseguido. Indicar a(s) razão(ões).

5. Epílogo

Incluir uma reflexão crítica sobre a disciplina (pontos a manter, a alterar e a eliminar).

Apêndice

Scripts (criar bd, criar tabelas e restrições, dados iniciais)

5. Notas finais

- 3) Embora a componente prática da disciplina corresponda a sete valores na nota final, individualmente cada aluno pode ter uma classificação diferente.

Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se aferir a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida dentro do grupo de trabalho. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelos docentes.

- 4) O SGBD a usar é o *SQL Server 2012 Express Edition* (ou outro, desde que aceite pelos docentes).

As aplicações devem ser desenvolvidas em java - *NetBeans* (ou outra, desde que aceite pelos docentes).

Ano lectivo 2016/2017

1. Enquadramento “Gestão de saúde”

(Notas: descrição da autoria de Dmytro Vasyanovych, monitor do DI-UBI)

Uma certa região tem 5 Hospitais que disponibilizam centro de saúde (CS), centro de internamento (CI), centro de análises (CA), centro de tratamento crônico (CT), centro operatório (CO), centro de maternidade (CM) e centro farmacêutico (CF) em cada um deles para a população das cidades próximas.

Foi pedido pela Administração regional da saúde informatizar os Hospitais.

Nos CS é necessário gerir os pacientes que vêm pedir tratamentos, é necessário marcar as consultas. Nas consultas deve ser possível visualizar o registo clínico do paciente das interacções com o centro de saúde e consequentemente com todos outros centros onde o paciente registou algum tipo de atividade, depois de fazer diagnóstico preliminar deve ser possível marcar encaminhamento do paciente para um dos centros mais adequados. Deve ser possível efectuar seguimento do diagnóstico preliminar gerando diagnósticos posteriores. Deve ser possível encaminhar o paciente para efectuar análises. Deve ser possível prescrever os medicamentos adequados, deve ser possível prescrever alguns dos medicamentos para tratamento de sintomas com base no diagnóstico anterior. Os diagnósticos anteriores podem ser acedidos pelo certo doutor somente depois do fornecimento de direitos de acesso pelo doutor anterior, quanto por este for solicitado visualização de registos protegido.

o CF tem os medicamentos, fisicamente localizados, deve haver uma gestão rigorosa dos medicamentos a entrar e a sair, deve se manter uma quantidade de medicamentos em armazenamento entre quantidades superior e inferior, indicadas pelo director do hospital. Deve haver medicamento que só podem ser prescritos por certo grupo de doutores. Deve haver maneira de visualizar todos os medicamentos presentes as suas quantidades existentes e quantidades usadas em certo período.

O encaminhamento de um paciente deve gerar um pedido de intervenção/ internamento em centro adequado, como marcação de actividade (operação, análise, internamento, tratamento crônico).

Internamentos ou tratamentos crônicos devem gerar o pedido de atribuição de uma cama/ maca em CI ou CT, quando o paciente chegar será lhe atribuída uma cama, ou lugar temporário indicando o local onde ficara o tal lugar temporário, deve ser possível admitir mais doentes do que camas disponíveis, é importante emitir alertas quando alguma cama fica disponível para limpeza/ preparação, é importante emitir alertas quando alguma cama ficar pronta para receber pacientes novos ou pacientes de lugares temporários.

No CO deve haver gestão rigorosa das salas operatórias, deve haver lista de operações feitas junto com as informações dos doutores e enfermeiros que estavam presentes na operação do paciente. Deve ser possível a visualização das salas operatórias em ocupação. O doutor que faz diagnóstico deve poder marcar a sala de operações para algum paciente com urgência no tratamento. Cada sala de operação tem equipamentos associados a sala, deve saber se quantidade de vezes que um certo equipamento foi usado, e os que tem validade temporal, devem ser emitidas as alertas para substituição
No CA deve ser possível visualizar as análises requeridas, as análises em produção e as análises completas, as análises completas devem ficar anexos aos registos clínicos dos Pacientes gerar alerta para o doutor que as pediu.

O CM tem as camas de espera, as salas de parto e camas pós-parto, deve ser possível visualizar as ocupações e gerar alertas quando cama estará disponível, prontas para próxima paciente.

2. Tarefas a realizar

2.1 Elaborar o modelo de dados para o sistema descrito.

Notas:

1. Assumir para as situações não especificadas as soluções que pareçam mais plausíveis. Indicar explicitamente as opções tomadas.
2. O modelo deve estar normalizado (3FN).

2.2 Base de dados

Implementar a base de dados em SQL Server 2014 (ou outro, desde que aceite pelo docente).

Escrever scripts para:

- Criar a base de dados;
- Criar tabelas e restrições tendo em conta o modelo de dados desenvolvido;
- Inserir alguns dados iniciais (dados de arranque).

Notas:

- 1) os **scripts devem ser escritos manualmente**, i.e., não se pretende a sua extração a partir do software (SQL Server Management Studio).
- 2) Os dados iniciais devem ser suficientes para a demonstração das funcionalidades das aplicações a serem desenvolvidas (10 linhas de dados para cada tabela – usada nas aplicações).

2.3 Aplicações

A. Desenvolver a aplicação “Agendar Consulta” para marcar uma consulta.

A aplicação cliente/servidor deve permitir:

- f) Visualizar marcações existentes (para um dado dia, médico Z, centro X do hospital Y).
- g) Agendar uma consulta (para uma dada hora/posição do dia).
- h) Visualizar o horário do médico para uma dada semana.

B. Desenvolver a aplicação “Consulta” para registar uma consulta de um paciente.

A aplicação cliente/servidor deve permitir:

- a) Visualizar dados de consultas anteriores (do paciente) e dos seus elementos conexos (análises e exames).
- b) Inserir os dados relevantes da consulta (sintomas, diagnóstico, prescrição de medicamentos/exames/análises, dados de exames/análises externos, etc.).

C. Desenvolver a aplicação “Agendar Exame” para agendar um exame.

A aplicação cliente/servidor deve permitir:

- a) Visualizar marcações existentes (para um dado dia no centro X do hospital Y).
- b) Agendar um exame.

D. Desenvolver a aplicação “Realizar Exame” para registar os dados de exame (ex., o relatório).

A aplicação cliente/servidor deve permitir:

- a) Registar dados do exame

E. Desenvolver a aplicação “Agendar Internamento” para agendar um internamento.

A aplicação cliente/servidor deve permitir:

- a) Visualizar os agendamentos existentes
- b) Agendar um internamento num dado quarto.

Notas finais:

- 1) As aplicações não necessitam de ambiente gráfico (janelas, diálogos, etc.).
- 2) Podem ser desenvolvidas outras funcionalidades. Contudo, **não serão valorizadas e se estiverem erradas descontam**. Por exemplo, não é pedida qualquer funcionalidade para registar pacientes/utentes.

2.4 Relatório

Elaborar um relatório com a descrição permonorizada do trabalho realizado.

3. Elementos a entregar, datas e cotação

3.1 Datas importantes

Modelo de dados, Scripts, Relatório e aplicação: 04 de junho de 2017, até às 24:00.

Discussão: semana 05-09 de junho de 2017.

3.2 Elementos a entregar

- Relatório em formato digital (.pdf e .doc)
- Scripts em formato texto (.txt ou .sql). Os scripts devem ser escritos manualmente, tal como usados nas aulas práticas. A entrega dos scripts obtidos diretamente a partir do SGBD será penalizada.
- *Source code* das aplicações

...

4. Estrutura do Relatório

O relatório deve conter, pelo menos, os seguintes capítulos:

Capa

Incluir a identificação do turno prático e dos elementos do grupo.

1. Introdução

Introdução ao trabalho desenvolvido e introdução genérica sobre as ferramentas utilizadas.

2. Modelo de dados e *scripts*:

Fazer uma breve introdução à modelação de dados e à construção do modelo conceptual.
Apresentar e justificar o modelo de dados desenvolvido.

Deve incluir: uma descrição da organização (tal como entendida pelo grupo); as opções tomadas para as situações não especificadas no enunciado; indicar as “regras de negócio” da organização.

Deve incluir a imagem (foto) dos modelos de dados produzidos durante as aulas (incluindo DEA assinado pelo docente do turno prático).

3. Aplicação

3.1 Decomposição e distribuição de tarefas

Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.

3.2 Acesso à base de dados

Documentar, devidamente, a forma como foi efectuado o acesso à base de dados. Incluir secções de código ilustrando o acesso à base de dados (exemplos para operações de consulta, inserção, eliminação e actualização).

3.3 Funcionalidade

Descrição da funcionalidade global, incluindo uma representação esquemática de como funciona a solução.

Incluir uma visita guiada (ecrãs ilustrativos e respetivos efeitos na BD) contendo:

Marcação de consulta → Realização de consulta com o registo das queixas/sintomas e diagnóstico → Marcação de exames/análises complementares → Realização/registo de exame/análises → Agendar internamento.

4. Conclusões

Indicar o que foi conseguido.

Indicar o que não foi conseguido. Indicar a(s) razão(ões).

5. Epílogo

Incluir uma reflexão crítica sobre a disciplina (aspetos a manter, a alterar e a eliminar).

Apêndice

Scripts (criar bd, criar tabelas e restrições)

5. Notas finais

- 1) Embora a componente prática da disciplina corresponda a sete valores, cada aluno pode ter uma classificação diferente.

Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se aferir a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida dentro do grupo de trabalho. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelos docentes.

- 2) O SGBD a usar é o *SQL Server 2014 Express Edition* (ou outro, desde que aceite pelos docentes).

- 3) As aplicações devem ser desenvolvidas em java - *NetBeans*.

Ano lectivo 2017/2018

1. GISS - Gestão Integrada de Serviços de Saúde

O país encontra-se dividido por Regiões de Saúde. As regiões de saúde podem conter vários Centros Hospitalares. Os serviços de saúde de cada centro hospitalar são geridos de modo centralizado, permitindo, por exemplo, marcar consultas de forma integrada em qualquer um dos seus hospitais.

Os centros hospitalares possuem diversos serviços: Centro de Saúde (CS), Centro de Internamento (CI), Centro de Análises (CA), Centro de Tratamento Crónico (CT), Centro Operatório (CO), Centro de Maternidade (CM), Centro Farmacêutico (CF), Centro de Urgência (CU), entre outros. Note-se que as mesmas valências podem existir em diversos hospitais do mesmo centro; no entanto, nem todos os serviços têm de existir em todos os centros hospitalares. Por questões funcionais, as diversas valências encontram-se organizadas por “Áreas Clínicas”: Cardiologia, Pediatria, Ortopedia, Urologia, etc.

Cada centro hospitalar tem o seu conjunto de colaboradores: médicos, enfermeiros, assistentes operacionais, entre outros. Os colaboradores têm um conjunto de habilitações académicas e podem possuir várias especializações. Cada área clínica tem o seu corpo clínico. Cada colaborador tem o seu horário de trabalho, podendo este distribuir-se por um ou mais serviços do centro hospitalar.

Os utentes têm de registar-se antes de poderem usufruir dos serviços de saúde — o registo faz-se na primeira interação com o centro hospitalar. Todas as interações dos utentes com os serviços do centro hospitalar são registadas.

Nos serviços de urgência os utentes são admitidos sem marcação prévia. Nos restantes serviços a admissão de utentes passa sempre por marcação. Por exemplo, as consultas externas carecem sempre de marcação prévia (que pode ser realizada por iniciativa do paciente, ou através de outros serviços de saúde).

A gestão integrada dos serviços de saúde pressupõe que os clínicos podem aceder ao histórico clínico dos pacientes. Durante as consultas, e após o diagnóstico preliminar, é possível prescrever medicamentos/exames/ análises clínicas e encaminhar os pacientes para outras áreas clínicas. Note-se também que uma consulta pode ir progredindo: atendido → diagnóstico → análises → exames → atendido → análises → atendido.... Alguns tratamentos, devido à sua especificidade, requerem recursos especiais, tais como camas ou macas.

O agendamento de internamentos requer a reserva de uma cama num determinado serviço de internamento (Medicina, Cirurgia, Cuidados Intensivos, etc.). Posteriormente, aquando do internamento, havendo disponibilidade é atribuída uma cama num quarto do correspondente serviço. Não havendo, poderá ser atribuído uma cama temporária noutro serviço, ou mesmo uma maca. É possível admitir mais doentes do que camas disponíveis, donde, é importante conhecer quando alguma cama (ou maca) fica disponível, para limpeza/preparação, ou pronta para receber pacientes.

A disponibilidade das salas de operações também tem de ser gerida com todo o rigor. Cada sala de operações tem o seu conjunto de equipamentos/utensílios. Cada equipamento tem a sua validade temporal, devendo, também, ser mantido um registo do número de vezes que foi usado. Cada sala tem associado um horário para marcação de intervenções cirúrgicas. Para cada intervenção cirúrgica é registado o paciente, o tipo de intervenção efetuada, a equipa clínica envolvida, entre outros elementos.

As análises clínicas e os exames, à semelhança das consultas, são realizados por pessoal com a competência adequada. Estes elementos auxiliares de diagnóstico possuem um estado: Concluído, Em Progresso, A aguardar. Os resultados são incluídos no respetivo processo (de diagnóstico) à medida que vão sendo obtidos.

O centro de maternidade tem recursos diversos, tais como Sala de Espera, Salas de Parto e Quarto pós-parto, que devem ser geridas de forma criteriosa.

2. Tarefas a realizar

2.1. Elaborar o modelo de dados (DEA + Esquema Relacional) para o sistema descrito.

Notas:

1. Para as situações não especificadas devem assumir-se as soluções que pareçam mais plausíveis. Indicar explicitamente as opções tomadas.
2. O modelo deve estar normalizado (3FN).

2.2. Base de dados

Implementar a base de dados sobre um SGBD Relacional Cliente/Servidor.

Escrever scripts para:

- Criar a base de dados;
- Criar tabelas e restrições tendo em conta o modelo de dados desenvolvido;
- Inserir alguns dados iniciais (dados de arranque).

Notas:

- 3) Os **scripts devem ser escritos manualmente**, i.e., não se pretende a sua extração a partir do software (SQL Server Management Studio).
- 4) Os dados iniciais devem ser suficientes para a demonstração das funcionalidades das aplicações a serem desenvolvidas (10 linhas de dados para cada tabela usada nas aplicações).

2.3 Aplicações em ambiente gráfico

A. Desenvolver a aplicação “Marcações” para efetuar agendamentos. A aplicação deve permitir:

- i) Visualizar horário semanal (de recursos humanos ou de recursos materiais).
- j) Efectuar um agendamento. Pode envolver um ou mais recursos humanos e/ou um ou mais recursos materiais.

Notas: 1) Considere que existem dois tipos de marcação: tipo lista/sequencial (por ordem de chegada) ou com entradas definidas (ex. períodos de 30 minutos).

2) Os horários de salas, cursos, alunos e professores pode ser uma boa metáfora.

B. Desenvolver a aplicação “Consulta” para registar o progresso de uma consulta. A aplicação deve permitir:

- c) Visualizar registo de consulta e o seu progresso. Deve permitir aceder/visualizar eventuais consultas anteriores e seus anexos (análises, exames e prescrições).
- d) Inserir os dados relevantes da consulta (sintomas, diagnóstico, prescrição de medicamentos/exames/análises, etc.).

C. Desenvolver a aplicação “Meios Complementares” para registar os dados de exame/análises. A aplicação deve permitir:

- b) Registar dados de exame/análise clínica. Deve permitir estabelecer ligação à respetiva consulta (caso exista).

D. Desenvolver a aplicação “Intervenção” para gerir as intervenções cirúrgicas. A aplicação deve permitir:

- e) Introduzir dados sobre a intervenção.
- f) Visualizar dados de uma intervenção.

Nota: O agendamento da intervenção é para ser efetuado a partir da aplicação Marcações.

Notas finais:

- 3) As aplicações não necessitam de ambiente gráfico (janelas, diálogos, etc.).
- 4) Podem ser desenvolvidas outras funcionalidades. Contudo, **não serão valorizadas e se estiverem erradas descontam**. Por exemplo, não é pedida qualquer funcionalidade para registar pacientes/utentes.

2.4 Relatório

Elaborar um relatório com a descrição pormenorizada do trabalho realizado.

3. Elementos a entregar, datas e cotação

3.1 Datas importantes

Modelo de dados, Scripts, Relatório e aplicação: 03 de junho de 2018, até às 23:59.

Discussão: semana 04-08 de junho de 2018, de acordo com o escalonamento feito pelo docente respetivo turno prático.

3.2 Elementos a entregar

- Relatório em formato digital (.pdf e .doc)
- Scripts em formato texto (.sql). Os scripts devem ser escritos manualmente, tal como usados nas aulas práticas. A entrega dos scripts obtidos diretamente a partir do SGBD será fortemente penalizada.
- *Source code* das aplicações.

...

4. Estrutura do Relatório

O relatório deve conter, pelo menos, os seguintes capítulos:

Capa

Incluir a identificação do turno prático e dos elementos do grupo.

1. Introdução

Introdução ao trabalho desenvolvido e introdução genérica sobre as ferramentas utilizadas.

2. Modelo de dados e *scripts*:

Fazer uma breve introdução à modelação de dados e à construção do modelo conceptual. Apresentar e justificar o modelo de dados desenvolvido.

Deve incluir: **uma descrição da organização (tal como entendida pelo grupo)**; as opções tomadas para as situações não especificadas no enunciado; indicar as “regras de negócio” da organização.

Deve incluir as imagens (foto/scan) dos modelos de dados produzidos durante as aulas (incluindo DEA assinado pelo docente do turno prático).

3. Aplicação

3.1 Decomposição e distribuição de tarefas

Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.

3.2 Acesso à base de dados

Documentar, devidamente, a forma como foi efectuado o acesso à base de dados. Incluir secções de código ilustrando o acesso à base de dados (exemplos para operações de consulta, inserção, eliminação e actualização).

3.3 Funcionalidade

Descrição da funcionalidade global, incluindo uma representação esquemática de como funciona a solução.

Incluir uma visita guiada (ecrãs ilustrativos e respetivos efeitos na BD) contendo:

Mostrar horário de um médico para uma dada semana

Mostrar horário de uma sala de operações.

Mostrar dados de uma intervenção cirúrgica.

Marcação de consulta → Realização de consulta com o registo das queixas/sintomas e diagnóstico → Marcação de exames/análises complementares → Realização/registo de exame/análises → Revisitar o Médico → Agendar nova consulta.

4. Conclusões

Indicar o que foi conseguido.

Indicar o que não foi conseguido. Indicar a(s) razão(ões).

5. Epílogo

Incluir uma reflexão crítica sobre a disciplina (aspectos a manter, a alterar e a eliminar).

Apêndice

Scripts (criar bd, criar tabelas e restrições, inserir dados)

5. Notas finais

- 1) Embora a componente prática da disciplina corresponda a sete valores, cada aluno pode ter uma classificação diferente.
Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se aferir a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida dentro do grupo de trabalho. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelos docentes.
- 2) O SGBD a usar é o *SQL Server 2014 Express Edition* (ou outro, desde que aceite pelo docente do turno prático).
- 3) As aplicações devem ser desenvolvidas em java - *NetBeans* (ou outra linguagem de programação, desde que aceite pelo docente do turno prático)

Ano lectivo 2019/2020

(Da autoria de Prof. Doutor Rui Cardoso)

Tema: Base de Dados dos serviços prestados pelo C4G

1. Introdução

O Colaboratório para as Geociências (C4G) (<https://www.c4g-pt.eu/pt/>) é uma infraestrutura de investigação distribuída dedicada às Ciências da Terra Sólida. Agrega 58 laboratórios de 15 instituições em Portugal. Tem como objetivo: partilhar o conhecimento, tecnologias, recursos e formação entre as instituições parceiras, e também com outras instituições públicas e privadas e utilizadores individuais que o solicitem. O C4G funciona como um balcão único de acesso para partilhar recursos que cada instituição parceira do C4G disponibiliza através dos seus laboratórios: dados, produtos, recursos humanos, equipamento e formação para ser utilizado nos serviços que o C4G disponibiliza.

A base de dados a implementar, destina-se a gerir a oferta dos serviços prestados pelo C4G à comunidade, incluindo o modo como se acede e partilha os recursos disponíveis. É necessário armazenar informação sobre: quem são os membros do C4G e também outros utilizadores externos que apenas utilizam os serviços. Os recursos podem ser pedidos aos laboratórios desde que disponíveis para serem incluídos no serviço específico. Os serviços podem ser solicitados por membros do C4G ou por outros. Os serviços, são supervisionados por grupos de trabalhos que atuam em várias linhas de ação e que caracterizam os vários tipos de serviços disponíveis no C4G. Cada serviço prestado tem sempre um responsável que é membro do C4G e um utilizador que o requere.

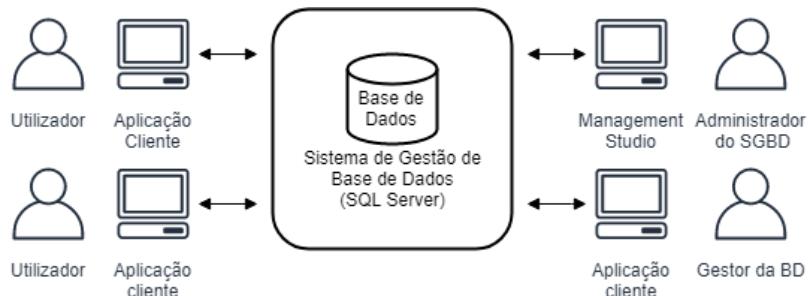


Figura 1 Exemplo de diferentes tipos de utilizadores/clientes que podem ter acesso à base de dados implementada no SGBD.

Existem membros do C4G que não pertencem às instituições parceiras, mas a outras externas. Uma instituição pode ser pública, privada ou um utilizador individual. Os membros do C4G e os outros utilizadores podem pertencer ou não a unidades de investigação. Existem também várias unidades de investigação que integram grupos de investigadores com interesses comuns. É também relevante mencionar que os diversos tipos de recursos têm características diferentes. Para concluir é necessário ainda registrar para cada serviço prestado uma lista discriminada dos recursos utilizados e a quem foi prestado. Podem existir vários tipos de utilizadores que podem aceder à base de dados, com níveis de acesso diferenciados e usando credenciais de acesso (user/password). Ter em atenção que os recursos são finitos e únicos e será necessário acautelar a verificação da sua disponibilidade e custos. No servidor web do C4G está mais informação disponível sobre o funcionamento, recursos e serviços prestados (<https://www.c4g->

<pt.eu/pt/>). Para esclarecimentos adicionais sobre a implementação e informação relevante não mencionada nesta especificação, para além do docente responsável rcardoso@ubi.pt, podem consultar também o Prof. Rui Fernandes rui@segal.ubi.pt e o Especialista de Informática Luís Carvalho luis.carvalho@c4g-pt.eu.

Na Fig. 2 são apresentados diferentes tipos de operações às quais a base de dados deve dar resposta.

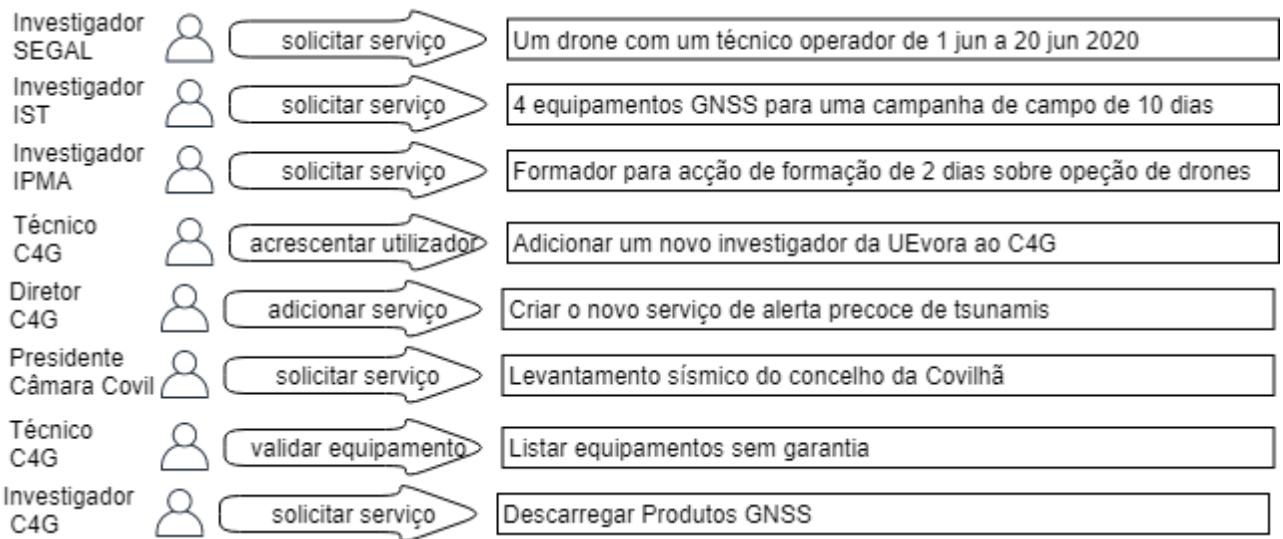


Figura 2 Exemplos de situações operacionais.

2. Objetivos do trabalho

1. Criar uma base de dados que permita gerir os recursos e serviços disponibilizados pelo C4G, preparada para satisfazer as operações mais relevantes para os diferentes tipos de utilizadores.
2. Efetuar a implementação de uma aplicação cliente para aceder à base de dados.
3. Escrever um relatório sobre o trabalho realizado.

3. Competências a adquirir

1. Desenvolver a capacidade de análise e de implementação de bases de dados.

4. Datas Importantes

Data de entrega: segunda-feira, 1 de junho de 2020 até às 13h.

Data da defesa: 1 a 5 de junho de 2020.

5. Grupos

O trabalho é realizado por grupos de 3 alunos, preferencialmente.

6. Avaliação

O trabalho está cotado para 5 valores.

Os componentes avaliados são: o modelo entidade relacionamento e o respetivo modelo relacional, o modelo de dados implementado, as funcionalidades implementadas na aplicação, o relatório e a apresentação.

O trabalho será defendido pelo grupo na última semana de aulas no turno respetivo, ou em horário a combinar com o docente da aula prática. Alunos que não participem na apresentação do trabalho terão zero valores. Cada aluno pode ter uma classificação diferente da dos colegas de grupo, refletindo deste modo o seu desempenho no trabalho.

e na discussão do mesmo. Todos os trabalhos serão demonstrados e defendidos perante o docente da disciplina em sessões de defesa do trabalho específicas para cada grupo. As defesas dos trabalhos têm duração aproximada de 15 a 20 minutos.

8. Tarefas a efetuar

No desenvolvimento da base de dados pode usar para a criação dos diagramas o [DBSchema](#) ou [draw.io](#), ou o [ERDPlus](#) para criar o diagrama de entidade relacionamento, o modelo relacional e o código SQL necessário. Em alternativa.

O sistema de gestão de base de dados para realizar o trabalho é o SQL Server.

A base de dados deve ser implementada através de um script SQL.

Deve criar um backup da base de dados com o modelo de dados e com dados inseridos, que permita depois reconstruir a base de dados em qualquer máquina com o SQL Server.

A base de dados deve estar preparada para dar resposta às questões mencionadas na introdução através de: consultas, inserções atualizações e remoções de dados para os vários tipos de utilizadores.

Ao derivar o diagrama entidade relacionamento e o correspondente esquema relacional tenha presente que as tabelas derivadas do modelo relacional devem estar normalizadas, assumindo para as situações não especificadas as soluções que pareçam mais plausíveis. No desenvolvimento deve também indicar explicitamente as escolhas efetuadas.

Desenvolver a aplicação cliente que permita interagir diretamente com a base de dados. No desenvolvimento da aplicação cliente podem usar a ferramenta que considerarem mais adequada. Em baixo está uma lista com várias alternativas de ferramentas e exemplos de aplicações clientes desenvolvidas em Java, FreePascal, C#, Python e Radzen para SQL Server (no trabalho tem de usar o SQL Server como SGBD).

- Exemplo aplicação JBuilder/SQL Server
http://www.di.ubi.pt/~pprata/bd/BD0405-SQLServer_JBuilder.pdf
- Exemplo aplicação JBuilder/SQL Server
<http://www.di.ubi.pt/~pprata/bd/Projeto.rar>
- Exemplo aplicação C#/SQL Server (<https://www.guru99.com/c-sharp-access-database.html>)
- Exemplo aplicação Lazarus/SQL Server
(<http://webx.ubi.pt/~rcardoso/bd1920/software/Ex2.zip> ou Moodle)
- Desenvolvimento de aplicações com Lazarus -
https://wiki.freepascal.org/Lazarus_Tutorial
- Exemplo aplicações Radzen (<https://demos.radzen.com/dashboard>)
- Exemplo desenvolvimento aplicação Radzen
<https://github.com/akorchev/radzen/tree/master/CRMDemo>
- Desenvolvimento em Azure/SQL Server (<https://docs.microsoft.com/pt-pt/azure/sql-database/sql-database-develop-overview>)
- Desenvolvimento em Azure/SQL Server (<https://sqlchoice.azurewebsites.net/en-us/sql-server/developer-get-started/>)

Elaborar um relatório descrevendo o trabalho realizado.

9. Documentação a entregar

Código em SQL para reconstruir a base de dados com dados no SGBD, o código da aplicação desenvolvida e o relatório.

Submeter através da página da disciplina no Moodle três ficheiros **x_y_z_bd.zip** com a base de dados, **x_y_z_app.zip** com a aplicação e **x_y_z_rel.zip** com o relatório (substituir x, y e z pelos números dos elementos do grupo e submeter dentro do prazo estabelecido).

10. Relatório

Deve ser elaborado um relatório detalhado abordando, pelo menos, os seguintes tópicos:

1. Introdução – Apresentação do trabalho desenvolvido.
2. Modelo de dados – Apresentação e justificação do modelo de dados desenvolvido, incluindo as opções tomadas para as situações não especificadas.
3. SGBD e Base de Dados – Apresentar o modelo relacional, as funcionalidades implementadas para gerir as interrogações à base de dados com exemplos.
4. Aplicação – Apresentar a aplicação e as funcionalidades implementadas.
5. Testes – Apresentar os testes que efetuou e os resultados obtidos.
6. Conclusão - Indicar o que foi conseguido. Indicar o que não foi conseguido e apresentar a(s) justificação(ões).

Não existe um template predefinido para o relatório, mas aconselha-se a utilização do template em Latex da disciplina de projeto <https://www.di.ubi.pt/~inacio/projeto/Formato-LaTeX.zip> e o editor <https://www.overleaf.com/login>.

11. Prémio

Como prémio pelo trabalho realizado, o SEGAL (Space & Earth Geodetic Analysis Laboratory <http://segal.ubi.pt/>) vai atribuir uma Bolsa de Iniciação à Investigação com a duração de 3 meses, no valor total de 1236€ (412€/mês) a um aluno do grupo com a melhor nota no trabalho prático. Os critérios de atribuição são os seguintes: ter tido a melhor nota no trabalho e esta tenha sido no mínimo 15 valores. Em caso de empate, será selecionado o aluno que tenha obtido melhor classificação à disciplina. Se a situação de empate persistir serão os docentes da disciplina a desempatar. A bolsa é atribuída ao aluno no próximo ano letivo para fazer o projeto final de licenciatura no SEGAL. Em caso de não aceitação pelo candidato classificado em primeiro lugar, a bolsa será atribuída ao primeiro aluno que a aceite entre os que tiveram a melhor nota no trabalho, podendo, caso os docentes assim o considerem, ser atribuída a um aluno de outro grupo (desde que a nota do trabalho tenha sido superior a 15 valores).

12. Referências

SQL Server 2000 Para Profissionais, Orlando Belo, FCA ISBN 972-722-505-5

SQL - Structured Query Language, Luís Manuel Dias Damas, FCA ISBN 972-722-443-1

Desenvolvimento de aplicações em SQL Server - <https://www.mssqltips.com/sql-server-tip-category/158/application-development/>

Documentação sobre Lazarus - https://wiki.freepascal.org/Lazarus_Documentation

Documentação sobre Radzen - <https://www.radzen.com/documentation/>

Desenvolvimento em Azure <https://sqlchoice.azurewebsites.net/en-us/sql-server/developer-get-started/>

Ano lectivo 2020/2021

1. Zurrapa – Drinks & Coffee

A empresa *Zurrapa-Drinks & Coffee, Lda.* dedica-se exclusivamente ao comércio de bebidas, sandes, bolos e outros produtos “leves”, tais como pastilhas e chocolates, em estabelecimentos de grandes dimensões (universidades, hospitais, etc.). Suponha que no próximo ano, a *Zurrapa* vai gerir/explorar os bares da UBI, onde pretende implementar serviço ao balcão, sala e em esplanada, se tiver condições para isso.

A *Zurrapa* definiu que todos os produtos são encaminhados a partir de um ponto de distribuição, designado por Armazém, para os diversos bares.

A empresa tem vários empregados. Cada empregado está atribuído a um bar, que corresponde ao seu local de trabalho pré-definido, e pode assumir um de dois papéis: empregado de balcão ou empregado de mesa. Na gestão diária da empresa, os empregados podem ser deslocados do seu local de trabalho pré-definido para outro bar ou para o Armazém em função das necessidades (por exemplo bares com mais procura à hora do almoço necessitam de mais funcionários). Cada bar tem um só responsável, que pode variar ao longo do ano.

Os empregados de mesa aceitam os pedidos dos clientes. Por questões de logística, o empregado de mesa que aceita o primeiro pedido para uma mesa, fica responsável pela mesma. Posteriormente, outros empregados podem aceitar pedidos para a mesma mesa.

O empregado de balcão consulta os pedidos “em aberto”, e, em função deles, prepara os produtos solicitados e entrega-os ao empregado de mesa, ficando o pedido marcado como satisfeito. Quando o serviço é pago, o pedido é fechado.

A recepção de mercadorias no Armazém, o seu armazenamento por categorias e posterior distribuição pelos bares é feita pelo empregado de Armazém. A *Zurrapa* tem por prática seleccionar, de modo aleatório, o empregado de Armazém somente entre os seus empregados de mesa (ou seja, os empregados de balcão não prestam serviço no Armazém). O escalonamento deste empregado realiza-se à sexta-feira e é válido para a semana seguinte.

No Armazém então depositados, sobretudo, produtos agregados (caixas e/ou embalagens), enquanto nos bares estão produtos individuais. Por exemplo, no armazém encontram-se grades de cerveja (contendo 24 garrafas), embalagens com 6 garrafas 1.5L de água, sacos com 1 Kg de café (correspondente a 60 cafés individuais), etc. Num bar podem encontrar-se 28 cervejas, 7 garrafas de água de 1.5L e “17” cafés. O empregado de Armazém é o responsável por verificar o stock dos bares e fazer a reposição dos produtos em falta.

Questões relevantes para a Gerência:

- a) É importante conhecer, a cada instante, o valor total (€) dos produtos existentes no Armazém e nos bares – estes valores reportam-se a preços de custo.
- b) É importante conhecer, para cada dia, quanto foi gasto (a preço de custo) e quanto foi recebido em cada bar.
- c) Deve ser desenvolvida uma aplicação que permita ao empregado de mesa “tratar” os pedidos dos clientes. No caso de os produtos solicitados não existirem em quantidade suficiente no bar, a aplicação deve indicar se esses

produtos existem em armazém e, nesse caso, quanto tempo demora a ser reposto o stock. Note-se que o empregado de mesa é o responsável por abrir/fechar as mesas.

- d) Deve ser desenvolvida uma aplicação para o empregado de balcão “tratar” os pedidos. Esta aplicação é responsável por actualizar o stock do bar à medida que o empregado satisfaz os pedidos.
- e) Deve ser desenvolvida uma aplicação para o Armazém que apresente as seguintes funcionalidades: carregar os stocks do armazém; consultar/carregar os stocks dos bares; produzir dados estatísticos para a Gerência.

2. Tarefas a realizar

2.1. Modelo Conceptual

Elaborar o Modelo Conceptual para o sistema descrito, usando para o efeito a notação introduzida nas aulas (teóricas).

Para as situações não especificadas devem assumir-se as soluções que pareçam mais plausíveis. Indicar explicitamente as opções tomadas.

2.2. Modelo Lógico

Producir o Modelo Lógico a partir do modelo conceptual e elaborar o esquema relacional (normalizado em 3FN).

2.3. Base de dados

Promover uma segunda passagem pelo modelo lógico (e esquema relacional) para se assegurar que todos os aspectos descritos no enunciado estão cobertos. Efetuar as alterações que sejam necessárias e justificar as opções tomadas.

Implementar a base de dados sobre um SGBD Relacional Cliente/Servidor.

Escrever scripts para:

- Criar a base de dados;
- Criar tabelas e restrições tendo em conta o modelo de dados desenvolvido;
- Inserir alguns dados iniciais (dados de arranque).

Notas:

- 5) Os **scripts devem ser escritos manualmente**, i.e., não se pretende a sua extração a partir do software (*SQL Server Management Studio*).
- 6) Os dados iniciais devem ser suficientes para a demonstração das funcionalidades das aplicações a serem desenvolvidas (10 linhas de dados para cada tabela usada nas aplicações).

2.3 Aplicações em ambiente gráfico

A. Desenvolver a aplicação Trata Pedido – Mesa (alínea c).

Cenários a considerar:

- 1) Dois pedidos, onde o stock do bar é suficiente para satisfazer cada um dos pedidos individualmente, mas não os dois em simultâneo.
- 2) Pedido de um produto com stock insuficiente no bar, mas com stock em armazém.
- 3) Três pedidos para mesma mesa, com identificação do responsável de mesa.
- 4) Consultar estado da mesa: despesa paga ou não-paga.
- 4) Marcar mesa livre/limpa.

B. Desenvolver a aplicação Trata Pedido – Balcão (alínea d).

Cenários a considerar:

- 1) Stock suficiente ($\text{qtd pedida} = \text{qtd servida}$).
- 2) Stock insuficiente ($\text{qtd pedida} \neq \text{qtd servida}$).
- 3) Pedido pago ($\text{qtd paga} = \text{qtd servida}$).
- 4) Pedido parcialmente pago ($\text{qtd paga} < \text{qtd servida}$).

C. Desenvolver a aplicação Trata Armazém (alínea e).

Cenários a considerar:

- 1) Actualizar stock armazém.
- 2) Consultar stocks bar (listagem).
- 3) Transferir stock armazém para bar.
- 4) Relatório com o valor dos stocks em armazém e bares (a preços de custo).
- 5) Relatório com a indicação, para cada dia, de quanto foi gasto (a preço de custo) e quanto foi recebido em cada bar. As datas de referência devem ser indicadas pelo utilizador.

2.4 Relatório

Elaborar um relatório com a descrição permonorizada do trabalho realizado.

3. Elementos a entregar, datas e cotação

(A detalhar em futuras versões)

Submeter no Moodle (poderá sofrer ligeiras alterações):

1. Constituição do grupo até 3^{af} 17 de Novembro.
2. Modelo de Dados (DEA), versão 1.0, até 3^{af} 24 de Novembro.
3. Modelos de Dados, versão 2.0, até 3^{af} 1 de Dezembro.
4. Modelo de dados, versão final, e Scripts SQL até 3^{af} 8 de Dezembro.
5. Aplicações até 3^{af} 15 de Dezembro.
6. Versão final, incluindo o relatório, até 3^{af} 5 de Janeiro.
7. Discussão, a partir de 6 de Janeiro.

4. Estrutura do Relatório

Capa

Imagen UBI/FE/DI

Licenciatura em...

UC: Bases de Dados

Título

Imagen ilustrativa/representativa do trabalho

Identificação dos elementos do grupo, ordenados por ordem crescente do número de estudante.

Local e data.

Agradecimentos

Incluir eventuais agradecimentos.

Resumo

Incluir um breve resumo (300 palavras).

Palavras-chave: 5 palavras ordenadas alfabeticamente.

Índice Geral

Lista de Abreviaturas

Lista de Fíguuras

Lista de Tabelas

1. Introdução

1.1 Enquadramento

1.2 Motivação

1.3 Objectivos

1.4 Organização do documento

2. Desenvolvimento de aplicações cliente/servidor sobre bases de dados

2.1 Introdução

2.2 Aplicações cliente/servidor

2.3 SQL Server

2.4. Configuração do acesso ao servidor

2.5 Lazarus (ou outro ambiente de desenvolvimento)

3. Modelação

3.1 Introdução (incluir a notação usada)

Fazer uma breve introdução à modelação de dados.

3.2 Descrição da organização

Indicar e justificar as opções tomadas para as situações não especificadas no enunciado.

3.3 Modelo Conceptual

Incluir também a imagem (foto) do modelo conceptual (rascunho manual,a lápis).

3.4 Modelo Lógico

Incluir também a imagem (foto) do modelo lógico (e esquema relacional).

3.5 Considerações

Eventuais considerações sobre o modelo final e sua implementação.

4. Aplicação

4.1 Distribuição de tarefas

Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.

4.1.xx Descrição precisa das tarefas (cada elemento do grupo faz a sua)

Cada elemento do grupo deve incluir uma subsecção com uma descrição muito precisa das suas tarefas. Pode incluir pseudo-código se tal for necessário. Se necessário, desenvolver nos apêndices.

4.2 Acesso à base de dados

Documentar, devidamente, a forma como foi efectuado o acesso à base de dados. Incluir excertos de código ilustrativos.

4.3 Funcionalidade

4.3.1 Descrição geral

Descrição da funcionalidade global, incluindo uma representação esquemática de como funciona a solução (BD; atendimento nas mesas, atendimento no balcão e interacção armazém-bares).

4.3.2 Aplicações

Descrição da funcionalidade sob os pontos de vista do utilizador (incluir imagens representativas – *screenshots*) e do programador (incluir excertos de código e SQL). Incluir os cenários descritos para cada aplicação.

- 4.3.2.1 Trata Pedido - Mesa
- 4.3.2.2 Trata Pedido - Balcão
- 4.3.2.3 Trata Pedido - Balcão

5. Conclusões

Indicar o que foi conseguido.
Indicar o que não foi conseguido. Indicar a(s) razão(ões).

Epílogo

Incluir uma reflexão crítica sobre a disciplina (pontos a manter, a alterar e a eliminar).

Referências Bibliográficas

Usar uma norma para as referências.

Anexos

Colocar aqui o que for acessório para a leitura do trabalho (não desenvolvido pelos autores).

Apêndices

Scripts (criar bd, criar tabelas e restrições, dados iniciais).
Outros elementos desenvolvidos pelos autores, mas acessórios para a leitura do trabalho.

Notas finais

1. Cada aluno pode ter uma classificação diferente.
Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se aferir a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida dentro do grupo de trabalho. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelos docentes.
2. O SGBD a usar é o *SQL Server* (ou outro, desde que aceite pelo docente do turno prático).
3. As aplicações devem ser desenvolvidas em *Lazarus* (ou outra linguagem de programação com capacidade de desenvolvimento RAD). Cada estudante deve desenvolver uma aplicação, devendo coordenar a sua actividade com os restantes membros da equipa.
4. Todos os estudantes têm de participar na produção do relatório.

Ano lectivo 2021/2022

1. Zurrapa – Drinks & Coffee

A empresa *Zurrapa-Drinks & Coffee, Lda.* dedica-se à exploração de snack-bares em estabelecimentos de grandes dimensões (universidades, hospitais, etc.). A empresa encontra-se sediada no nº 1 da Avenida Montes Hermínios, Covilhã, e possui filiais em vários pontos do país.

A Gerência da Zurrapa pretende implementar um sistema de informação que lhe permita conhecer as vendas nas suas filiais, nomeadamente, a situação de caixa e armazém. Apesar de ainda não ter uma ideia muito precisa como tal solução será implementada, a Gerência já decidiu que a sede e cada uma das suas filiais terá o seu servidor de bases de dados (SQL Server). Na sede a base de dados terá a designação de *ZurrapaSede*, nas filiais terá a designação *ZurrapaFilial*.

Na base de dados *ZurrapaSede* foi decidido criar a tabela *Filiais* para colocar os dados sobre as filiais (ID, Designação, Endereço, e-mail, telefone, gerente, ...), a tabela *FilialDia* para guardar os dados sobre os totais (€) recebidos e gastos, por dia, em cada filial e a tabela *FilialDiaBar* para colocar os dados sobre os totais recebidos e gastos em cada filial, por dia e por bar. O valor total recebido corresponde ao somatório dos valores entrados em caixa, o valor total de gastos corresponde ao preço de custo dos produtos vendidos (e que estiveram na origem do valor recebido).

As filiais têm um gerente, que é um empregado da filial, e funcionam sempre no modelo pré-pagamento. Em cada filial os produtos são sempre encaminhados a partir de um ponto de distribuição, designado por Armazém, para os diversos bares.

A empresa tem vários empregados. Cada empregado está atribuído a um bar, que corresponde ao seu local de trabalho pré-definido, e pode assumir um de dois papéis: empregado de balcão ou empregado de caixa. Na gestão diária da empresa, os empregados podem ser deslocados do seu local de trabalho pré-definido para outro bar ou para o Armazém em função das necessidades (por exemplo bares com mais procura à hora do almoço necessitam de mais funcionários). Cada bar tem um só responsável, que pode variar ao longo do ano.

O empregado de balcão consulta os pedidos “em aberto”, e, em função deles, prepara os produtos solicitados e coloca-os no balcão, ficando o pedido marcado como satisfeito. A limpeza de mesas, caso existam, também está a cargo dos empregados de balcão.

A recepção de mercadorias no Armazém, o seu armazenamento por categorias e posterior distribuição pelos diversos bares da filial é tarefa do empregado de Armazém. Este empregado é seleccionado de modo aleatório entre os seus empregados de balcão (ou seja, os empregados de caixa não prestam serviço no Armazém). Este sorteio realiza-se à sexta-feira e é válido para a semana seguinte.

No Armazém então depositados, sobretudo, produtos agregados (caixas e/ou embalagens), nos bares estão, sobretudo, produtos individuais. Por exemplo, no armazém encontram-se grades de cerveja (contendo 24 garrafas), embalagens com 6 garrafas 1.5L de água, sacos com 1 Kg de café (correspondente a 60 cafés individuais), etc. Num bar podem encontrar-se 28 cervejas, 7 garrafas de água de 1.5L e “17” cafés. O empregado de Armazém é o responsável por verificar o stock dos bares e fazer a reposição dos produtos em falta.

Para as filiais é importante conhecer:

- a) A situação corrente: o valor total (€) dos produtos existentes no Armazém e nos bares – estes valores reportam-se a preços de custo.
- b) Quanto foi gasto (a preço de custo) e quanto foi recebido em cada bar (por dia ou por intervalo de datas).

Para a sede é importante conhecer os valores totais recebidos e consumidos, por dia ou por intervalo de datas:

- c) Valores globais (somatório dos valores das filiais).
- d) Por filial.

Funcionalidades desejadas nas filiais:

- a) Caixa: aceita pedido do cliente. No caso de os produtos solicitados não existirem em quantidade suficiente no bar, a aplicação deve indicar se esses produtos existem em armazém e, nesse caso, quanto tempo demora a ser reposto o stock.
- b) Balcão: consultar o pedido do cliente e marca-o como satisfeito.
- c) Armazém: consulta/carrega os stocks do armazém e consulta/carrega os stocks dos bares.
- d) Fecho de caixa: marca o fim da operação do dia e carrega os dados para a sede. Esta operação é efectuada uma só vez por dia.
- e) Estatísticas: produz as estatísticas identificadas.

Funcionalidade desejada na sede:

- f) Estatísticas: consulta as estatísticas identificadas.

2. Tarefas a realizar

2.1. Diagrama Entidade-Associação e Esquema Relacional

Elaborar o Diagrama Entidade-Associação, considerando o Modelo Relacional, e o respectivo esquema relacional, usando para o efeito a notação introduzida nas aulas (teóricas).

Para as situações não especificadas devem assumir-se as soluções que pareçam mais plausíveis. Indicar explicitamente as opções tomadas.

2.2. Base de dados

Promover uma segunda passagem pelo modelo lógico (e esquema relacional) para se assegurar que todos os aspectos descritos no enunciado estão cobertos. Efectuar as alterações que sejam necessárias e justificar as opções tomadas.

Implementar a base de dados sobre um SGBD Relacional Cliente/Servidor.

Escrever scripts para:

- Criar a base de dados;
- Criar tabelas e restrições tendo em conta o modelo de dados desenvolvido;
- Inserir alguns dados iniciais (dados de arranque).

Notas:

- 1) Os **scripts devem ser escritos manualmente**, i.e., não se pretende a sua extracção a partir do software (*SQL Server Management Studio*).
- 2) Os dados iniciais devem ser suficientes para a demonstração das funcionalidades das aplicações a serem desenvolvidas (10 linhas de dados para cada tabela usada nas aplicações).

2.3 Aplicações em ambiente gráfico

A. Desenvolver a aplicação Caixa e Balcão (alíneas a) e b)).

Cenários a considerar:

- 1) Dois pedidos, onde o stock do bar é suficiente para satisfazer cada um dos pedidos individualmente, mas não os dois em simultâneo.
- 2) Pedido de um produto com stock insuficiente no bar, mas com stock em armazém.

B. Desenvolver a aplicação Trata Armazém (alínea c)).

C. Fecho de caixa (alínea d)).

D. Estatísticas filial (alínea e)).

E. Estatísticas sede (alínea f)).

2.4 Relatório

Elaborar um relatório com a descrição permonorizada do trabalho realizado.

3. Elementos a entregar, datas e cotação

(A desenvolver em futuras versões)

Submeter no Moodle:

4. Estrutura do Relatório

Capa

Imagen UBI/FE/DI

Licenciatura em...

UC: Bases de Dados

Título

Imagen ilustrativa/representativa do trabalho

Identificação dos elementos do grupo, ordenados por ordem crescente do número de estudante.

Local e data.

Agradecimentos

Incluir eventuais agradecimentos.

Resumo

Incluir um breve resumo (300 palavras).

Palavras-chave: 5 palavras ordenadas alfabeticamente.

Índice Geral

Lista de Abreviaturas

Lista de Fíguras

Lista de Tabelas

1. Introdução

1.1 Enquadramento

1.2 Motivação

- 1.3 Objectivos
- 1.4 Organização do documento
2. Desenvolvimento de aplicações cliente/servidor sobre bases de dados
 - 2.1 Introdução
 - 2.2 Aplicações cliente/servidor
 - 2.3 SQL Server
 - 2.4. Configuração do acesso ao servidor
 - 2.5 Lazarus (ou outro ambiente de desenvolvimento)

3. Modelação

- 3.1 Introdução (incluir a notação usada)
Fazer uma breve introdução à modelação de dados.
- 3.2 Descrição da organização
Indicar e justificar as opções tomadas para as situações não especificadas no enunciado.
- 3.3 Modelo de dados
Incluir imagens (foto) do modelo elaborado (rascunho manual,a lápis).
- 3.3 Considerações
Eventuais considerações sobre o modelo final e sua implementação.

4. Aplicações

4.1 Distribuição de tarefas

Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.

4.1.xx Descrição precisa das tarefas (cada elemento do grupo faz a sua)

Cada elemento do grupo deve incluir uma subsecção com uma descrição muito precisa das suas tarefas. Pode incluir pseudo-código se tal for necessário. Se necessário, desenvolver nos apêndices.

4.2 Acesso à base de dados

Documentar, devidamente, a forma como foi efectuado o acesso à base de dados. Incluir excertos de código ilustrativos.

4.3 Funcionalidade

4.3.1 Descrição geral

Descrição da funcionalidade global, incluindo uma representação esquemática de como funciona a solução.

4.3.2 Aplicações

Descrição da funcionalidade sob os pontos de vista do utilizador (incluir imagens representativas – *screenshots*) e do programador (incluir excertos de código e SQL). Incluir os cenários descritos para cada aplicação.

4.3.2.1 Aplicação 2.3 A

4.3.2.2 Aplicação 2.3 B

...

5. Conclusões

Indicar o que foi conseguido.

Indicar o que não foi conseguido. Indicar a(s) razão(ões).

Epílogo

Incluir uma reflexão crítica sobre a disciplina (pontos a manter, a alterar e a eliminar).

Referências Bibliográficas

Usar uma norma para as referências.

Anexos

Colocar aqui o que for acessório para a leitura do trabalho (não desenvolvido pelos autores).

Apêndices

Scripts (criar bd, criar tabelas e restrições, dados iniciais).

Outros elementos desenvolvidos pelos autores, mas acessórios para a leitura do trabalho.

5. Notas finais

(A desenvolver em futuras versões)

- 1) Cada aluno pode ter uma classificação diferente.
- 2) Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se aferir a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida dentro do grupo de trabalho. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelos docentes.
- 3) O SGBD a usar é o *SQL Server* (ou outro, desde que aceite pelo docente do turno prático).
- 4) As aplicações devem ser desenvolvidas em *Lazarus* (ou outra linguagem de programação com capacidade de desenvolvimento RAD). Cada estudante deve desenvolver uma aplicação, devendo coordenar a sua actividade com os restantes membros da equipa.
- 5) Todos os estudantes têm de participar na produção do relatório.

Ano lectivo 2022/2023 (EI)

1. Gestão de Projetos

O Departamento de Informática da Universidade da Beira Interior pretende implementar um sistema de informação que permita acompanhar as atividades associadas ao desenvolvimento de projetos de investigação e de prestações de serviço à comunidade.

Os projetos e os contratos de prestação de serviços têm um título, um nome curto, uma descrição e um estado (aprovado, cancelado, concluído, em curso, encerrado, renovado, em submissão) e datas de início e de fim.

Os projetos são sempre redigidos em português e em inglês; estão associados a um domínio científico e a uma área científica específica de aplicação (incluindo situações que abranjam múltiplas áreas); têm um conjunto de palavras-chave associadas (pelo menos 3). Alguns projetos possuem URL e/ou DOI.

Os projetos podem ter financiamento, interno ou externo que pode envolver outras entidades nacionais e internacionais (Estado Português, Fundação para a Ciência e a Tecnologia, Fundação Gulbenkian, Fundação "la Caixa" etc...) sendo também necessário armazenar informação sobre o ponto de contacto da entidade (nome, descrição, email, tel) e também a sua designação, sigla (ex: FCT), morada, URL e país.

Os projetos podem ser financiados através de um ou vários programas específicos (Portugal 2020, programa Europeu Horizon2020, EEA Grants etc..).

Em cada projeto existe um investigador responsável (IR) que deve exercer essa função a 35% do seu tempo. De modo a impedir atribuições indevidas, nenhum membro da equipa de investigação pode ficar com uma percentagem de afetação a projetos nacionais superior a 100%; sendo também considerada para os membros da equipa de investigação a percentagem mínima de tempo de dedicação de 15%.

Cada membro do DI pode ser promotor, copromotor, líder ou participante em projetos/contratos liderados ou não pela UBI. Sendo as suas atividades no projeto enquadradas através da unidade de investigação a que pertence, e que deve estar também representada no sistema de informação. Os membros do DI são identificados pelo seu número de funcionário e também por identificadores únicos como o ORCID.

Os projetos/contratos possuem um custo total elegível, discriminado por cada uma das unidades de investigação das várias instituições participantes.

As publicações resultantes do projeto devem ser associadas ao mesmo, através de DOI ou referência ao URL da publicação, e constituem um indicador de sucesso do projeto.

O sistema a implementar deve permitir:

- 1) Identificar a situação atual de cada um dos projetos com o detalhe adequado.
- 2) Para cada docente/investigador indicar a situação atual, discriminar as atividades em que está envolvido, nomeadamente mostrar o tempo de dedicação em cada uma e o tempo total.
- 3) Mostrar as áreas onde a investigação do DI está a incidir mais, ponderada pelo número de recursos humanos a ela dedicados.

- 4) Mostrar o peso relativo das principais fontes de financiamento dos projetos e serviços.
- 5) Classificar os projetos em função: do financiamento obtido, do número de docentes/investigadores participantes, do número de colaborações com instituições nacionais e internacionais, e também pelo número de publicações resultantes.
- 6) Indicar as principais colaborações com entidades externas.
- 7) Produzir estatísticas para melhor caracterizar as atividades desenvolvidas anualmente e por períodos mais longos.
- 8) Evitar situações anómalas (por ex.: incluir custos por instituição incompatíveis com o valor disponível e ou que ultrapassem o valor total, tempo de dedicação acima do limite definido).

Funcionalidades:

- a) Inserir projetos/serviços incluindo todos os elementos necessários (equipa de investigação, o investigador responsável e entidades externas participantes).
- b) Inserir entidades financiadoras e respetivos programas de financiamento.
- c) Alterar o estado de um projeto (por ex.: adicionar novo membro de equipa, e alterar o tempo de dedicação do IR).
- d) Mostrar a situação atual de um projeto/serviço.
- e) Mostrar a situação atual de um docente/investigador.
- f) Mostrar as estatísticas identificadas.
- g) Garantir a integridade e consistência dos dados.

2. Tarefas a realizar

2.1. Diagrama Entidade-Associação e Esquema Relacional

Elaborar o Diagrama Entidade-Associação, considerando o Modelo Relacional, e o respetivo esquema relacional, usando para o efeito a notação introduzida nas aulas (teóricas).

Para as situações não especificadas devem assumir-se as soluções que pareçam mais plausíveis. Indicar explicitamente as opções tomadas.

2.2. Base de dados

Promover uma segunda passagem pelo modelo lógico (e esquema relacional) para se assegurar que todos os aspectos descritos no enunciado estão cobertos. Efetuar as alterações que sejam necessárias e justificar as opções tomadas.

Implementar a base de dados sobre um SGBD Relacional Cliente/Servidor.

Escrever scripts para:

- Criar a base de dados;
- Criar tabelas e restrições tendo em conta o modelo de dados desenvolvido;
- Inserir alguns dados iniciais (dados de arranque).

Notas:

- 1) Os **scripts devem ser escritos manualmente**, i.e., não se pretende a sua extração a partir do software (*SQL Server Management Studio*).
- 2) Os dados iniciais devem ser suficientes para a demonstração das funcionalidades das aplicações a serem desenvolvidas (10 linhas de dados para cada tabela usada nas aplicações).

2.3 Aplicação em ambiente gráfico.

Desenvolver a aplicação GestaoProjetos.

- a. Implementar as operações CRUD necessárias (alíneas a) b) c) d) e)).

Cenário a considerar na implementação:

Inserir um novo projeto completo.

Mostrar a lista de projetos em curso por palavra-chave.

- b. Visualização das Estatísticas (alínea f)).

Cenário a considerar na implementação:

Mostrar publicações por projeto.

Mostrar participantes por projeto.

Mostrar projetos por financiamento.

- c. Assegurar consistência (alínea g))

Cenários a considerar:

Inserir novo projeto onde um docente/investigador tenha ultrapassado o limite de tempo de dedicação total.

Verificar fundos alocados por entidade participante.

2.4 Relatório

Elaborar um relatório com a descrição pormenorizada do trabalho realizado.

3. Elementos a entregar, datas e cotação

(A desenvolver em futuras versões)

Submeter no Moodle:

1. ...

4. Estrutura do Relatório

Capa

Imagen UBI/FE/DI

Licenciatura em...

UC: Bases de Dados

Título

Imagen ilustrativa/representativa do trabalho

Identificação dos elementos do grupo, ordenados por ordem crescente do número de estudante.

Local e data.

Agradecimentos

Incluir eventuais agradecimentos.

Resumo

Incluir um breve resumo (300 palavras).

Palavras-chave: 5 palavras ordenadas alfabeticamente.

Índice Geral

Lista de Abreviaturas

Lista de Fíguuras

Lista de Tabelas

1. Introdução

1.1 Enquadramento

1.2 Motivação

1.3 Objectivos

1.4 Organização do documento

2. Desenvolvimento da aplicação cliente/servidor sobre bases de dados

2.1 Introdução

2.2 Aplicação cliente/servidor

2.3 SQL Server

2.4. Configuração do acesso ao servidor

2.5 Lazarus (ou outro ambiente de desenvolvimento)

3. Modelação

3.1 Introdução (incluir a notação usada)

Fazer uma breve introdução à modelação de dados.

3.2 Descrição da organização

Indicar e justificar as opções tomadas para as situações não especificadas no enunciado.

3.3 Modelo de dados

Incluir imagens (foto) do modelo elaborado (rascunho manual,a lápis).

3.3 Considerações

Eventuais considerações sobre o modelo final e sua implementação.

4. Aplicação

4.1 Distribuição de tarefas

Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.

4.1.xx Descrição precisa das tarefas (cada elemento do grupo faz a sua)

Cada elemento do grupo deve incluir uma subsecção com uma descrição muito precisa das suas tarefas. Pode incluir pseudo-código se tal for necessário,que pode detalhar e desenvolver nos apêndices.

4.2 Acesso à base de dados

Documentar, devidamente, a forma como foi efectuado o acesso à base de dados. Incluir excertos de código ilustrativos.

4.3 Funcionalidade

4.3.1 Descrição geral

Descrição da funcionalidade global, incluindo uma representação esquemática de como funciona a solução.

4.3.2 Aplicação

Descrição da funcionalidade sob os pontos de vista do utilizador (incluir imagens representativas – *screenshots*) e do programador (incluir excertos de código e SQL). Incluir os cenários descritos para cada aplicação.

5. Conclusões

Indicar o que foi conseguido.

Indicar o que não foi conseguido. Indicar a(s) razão(ões).

Epílogo

Incluir uma reflexão crítica sobre a disciplina (pontos a manter, a alterar e a eliminar).

Referências Bibliográficas

Usar uma norma para as referências.

Anexos

Colocar aqui o que for acessório para a leitura do trabalho (não desenvolvido pelos autores).

Apêndices

Scripts (criar bd, criar tabelas e restrições, dados iniciais).

Outros elementos desenvolvidos pelos autores, mas acessórios para a leitura do trabalho.

5. Notas finais

(A desenvolver em futuras versões)

1. Cada aluno pode ter uma classificação diferente.
2. Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se aferir a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida dentro do grupo de trabalho. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelos docentes.
3. O SGBD a usar é o *SQL Server* (ou outro, desde que aceite pelo docente do turno prático).
4. As aplicações devem ser desenvolvidas em *Lazarus* (ou outra linguagem de programação com capacidade de desenvolvimento RAD). Cada estudante deve desenvolver uma aplicação, devendo coordenar a sua atividade com os restantes membros da equipa.

5. Todos os estudantes têm de participar na produção do relatório.

Ano lectivo 2023/2024 (EI)

1. Objetivos e competências

Objetivos: preparar os estudantes para entender, projetar e desenvolver soluções informáticas sobre bases de dados.

Competências: desenvolver modelos de dados; produzir esquema relacional; desenvolver aplicações multiutilizador sobre bases de dados cliente/servidor.

2. Contexto

A rede RADNET Portugal (<https://radnet.apambiente.pt/>) surgiu como a resposta nacional ao acidente de Chernobyl. Este acidente nuclear ocorreu a 26 de abril de 1986 na Ucrânia e libertou material radioativo para a atmosfera que afetou vários países europeus. A RADNET faz parte da rede europeia de monitorização da radioatividade e efetua em contínuo a monitorização da radioatividade ambiente. funciona também como sistema de alerta em caso de acidente nuclear ou emergência radiológica. Existem riscos radiológicos que podem afetar Portugal. Por exemplo, derivados das escórias das minas de urânio na Urgeiriça, do Reator Nuclear Português no Campus Tecnológico e Nuclear (CTN) do Instituto Superior Técnico (<https://tecnico.ulisboa.pt/pt/sobre-o-tecnico/campi/tecnologico-e-nuclear/>), riscos de incidentes em navios ancorados em portos nacionais ou em transito que tenham propulsão nuclear, lixo radioativo ou armas nucleares, mas também, riscos mais elevados derivados das centrais nucleares espanholas onde há 7 reatores nucleares ativos em 5 centrais.

Tabela 1: Efeitos da radiação

Dose (mSv)	Efeito
10000	É fatal em semanas.
6000	Foi a dose típica registada pelos trabalhadores da central nuclear de Chernobyl que morreram após um mês.
5000	É fatal em 50% das exposições no prazo de 1 mês.
1000	A radiação provoca enjoos e náuseas, mas não a morte.
400	Valor máximo/hora registado na central nuclear de Fukushima depois da explosão dos reatores.
350	Valores de exposição dos residentes de Chernobyl que foram relocados.
100	Valor máximo recomendado para trabalhadores expostos a radiação nas suas atividades em 5 anos.
10	Valor de exposição de um TAC (tomografia axial computorizada).
9	Valor de exposição da tripulação de voos entre Nova Iorque e Tóquio pela rota polar durante um ano.
0.4	Valor de exposição numa mamografia.
0.1	Valor de exposição num radiografia aos pulmões.
0.01	Valor de exposição numa radiografia dentária.

No passado já ocorreram diversos acidentes, um dos quais com libertação de radioatividade para o Tejo na central nuclear de Almaraz. Esta devia já ter sido encerrada por ter atingido o seu fim de vida útil, mas este foi recentemente aumentado para mais 20 anos e a posterior conversão do local num aterro de resíduos nucleares. No passado já houve tentativas de criar um aterro nuclear junto à fronteira nacional no norte perto fronteira do distrito de Bragança, no Douro e agora no Tejo junto à central de Almaraz, todos estes riscos constituem um perigo latente de poluição, que pode afetar a qualidade de vida, saúde e a economia nacional durante muitas gerações caso ocorra algum acidente grave. A exposição a radiação é medida em Sv (Sievert). Na tabela 1 mostramos uma relação entre a exposição e os efeitos produzidos. Relativamente aos riscos, a Agência Internacional da Energia Atómica ([IAEA](#) International Atomic

Energy Agency) classifica os riscos em três tipos principais: acidentes, incidentes e desvios de acordo com a informação na figura 1. As estações de vigilância portuguesas da rede RADNET estão implantadas estratégicamente junto à fronteira e nos principais núcleos populacionais e industriais. A localização das estações é a seguinte: Abrantes, Beja, Bragança, Castelo Branco, Coimbra, Elvas, Évora, Faro, Fratel (estaçao submersa no Tejo junto à fronteira), Funchal, Junqueira, Juromenha (estaçao junto ao Guadiana), Lisboa, Meimoa (na fronteira a 100 km da central nuclear de Almaraz com dois reatores ativos), Penhas Douradas, Pocinho (estaçao submersa no Douro junto à fronteira), Ponta Delgada, Portalegre, Porto e Sines e uma estaçao na base aérea de Talavera La Real em Badajoz (Espanha).

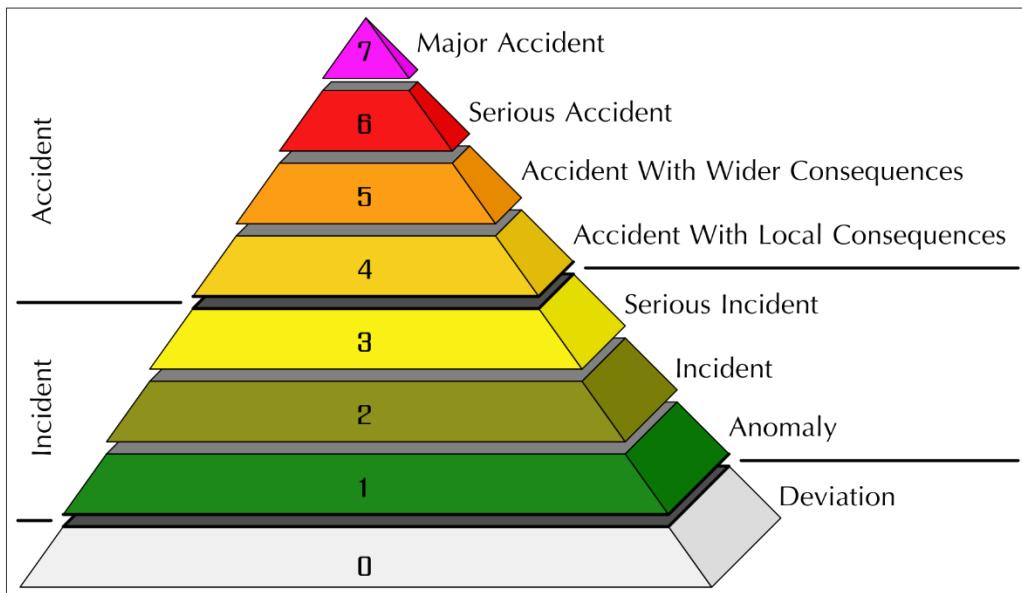


Figura 1: Classificação dos tipos de riscos, (fonte:
IAEA)

3. Descrição do trabalho

Pretende-se implementar um sistema de informação semelhante ao da RADNET portuguesa composto por uma base de dados e uma aplicação que permita interagir com o sistema desenvolvido. Para esse fim cada estação da rede possui ou pode possuir vários tipos de sensores de radiação no ar, na água e no solo. As estações podem ser permanentes (fixas) ou móveis e serem capazes de detetar radiação em várias gamas. Para cada estação deve ser indicada a sua localização e características. Nomeadamente, a data da instalação, o tipo e quantidade de sensores que possui e frequência de leituras. A unidade de medida mais utilizada para classificar o nível de radiação é o Sievert (Sv), na prática é usado o mSv como unidade de medida que corresponde a 0,001 Sv. Cada sensor tem um limite máximo e mínimo de sensibilidade. Pelo que, em algumas estações há sensores com diferentes graus de sensibilidade para procurar cobrir vários espectros de radiação. Todas as estações registam o nível de radiação em Sievert (Sv) com frequências distintas: à hora, ao minuto, ao segundo e possuem também valores de referência diários, mensais e anuais. É necessário que a base de dados mostre em tempo real as leituras nos diversos pontos da rede, permita também o acesso a dados históricos e que possa ativar e gerir alertas no caso dos valores ultrapassarem valores pré-definidos e ou de referência. Os níveis de alerta devem estar associados a uma gama de valores de radiação.

Cada estação fixa está equipada com pelo menos dois sensores/detectores Geiger-Müller, um para baixos níveis de radiação (de 10 nSv/h a 2 mSv/h) e outro para valores elevados (de 0,1 mSv/h a 10 Sv/h). É necessário registar também leituras com de estações móveis e de equipamentos portáteis que podem ser utilizadas em locais específicos durante vários períodos. Por exemplo, efetuar leituras nas minas da Urgeiriça durante 3 semanas. Com base nos valores detetados deve ser possível gerar alertas. Por

exemplo, se for detetado numa estação um valor 10% acima do valor médio anual, deve ser acionado um alarme e participada a situação à Agência Portuguesa do Ambiente e à Autoridade Nacional de Proteção Civil, se o valor detetado tem impacto direto na saúde o alerta deve incluir avisos diretos à população, por exemplo sirenes, altifalantes, SMS, chamadas telefónicas, TV, rádio, internet, email).

Deve também ser criada uma escala de valores admissíveis baseado na informação já existente e disponível para download das estações da rede RADNET.

4. Tarefas a realizar

4.1. Diagrama Entidade-Associação e Esquema Relacional

Elaborar o Diagrama Entidade-Associação em ERD Plus (<https://erdplus.com/>). Para as situações não especificadas devem assumir-se as soluções que pareçam mais plausíveis. Indicar explicitamente as opções tomadas.

Desenvolver o respetivo esquema relacional, com as relações normalizadas em 3FN.

4.2. Base de dados (em SQL Server)

- Gerar os scripts para criar a base de dados e as tabelas em SQL Server;
- Criar a base de dados e as tabelas a partir dos scripts;
- Importar para a base de dados os dados reais correspondentes às leituras do último ano, disponíveis para download no formato CSV para todas as estações permanentes em RADNET, dados adicionais, só os necessários para demonstrar as funcionalidades implementadas.

4.3 Aplicação em ambiente gráfico.

Criar uma aplicação (em *Python*) que permita testar as funcionalidades implementadas.

Use como “inspiração” a página da RADNET.

4.4 Relatório

Elaborar um relatório com a descrição permonorizada do trabalho realizado.

5. Datas importantes e elementos a entregar

- Trabalho Prático (*1^a entrega*): 17 de dezembro 2023, até às 23:59, submeter no moodle. Entrega DEA, esquema relacional e scripts.
- Trabalho Prático (*deadline*): 03 de janeiro 2023, quarta-feira, 23:59, submeter no moodle. Entrega: tudo.
- Trabalho Prático (defesa): nas aulas práticas das duas primeiras semanas de janeiro até dia 12 (a confirmar).
 - Datas das defesas:
 - PL1 segunda-feira 8 janeiro 2024 às 11h
 - PL2 quarta-feira 3 janeiro 2024 às 16h
 - PL2 quarta-feira 10 janeiro 2024 às 16h
 - PL3 quarta-feira 3 janeiro 2024 às 14h
 - PL3 quarta-feira 10 janeiro 2024 às 14h
 - PL4 sexta-feira 5 janeiro às 16h

- PL4 sexta-feira 12 janeiro às 16h
- PL5 sexta-feira 5 janeiro às 11h
- PL5 sexta-feira 12 janeiro às 11h

O nome dos ficheiros a submeter deve seguir o seguinte formato:

Turno_Grupo_NAluno1_NAluno2_NAluno3.XYZ, onde turno e grupo são a designação do turno e o número de grupo, NAluno representa o número de aluno e XYZ a extensão do ficheiro.

Exemplo: PL1_1_12345_67890_54321.sql.

Só um dos elementos do grupo é que submete os trabalhos. As submissões são feitas sempre pelo mesmo estudante.

Submissão 17 dezembro: ficheiro .pdf com o DEA e esquema relacional; ficheiro .sql com os scripts de criação das tabelas.

Submissão 03 de janeiro: relatório (.pdf); aplicação (.rar); scripts (.sql); e script para carregamento dos dados (.sql) ou ficheiro de backup da base de dados (.bak).

6. Estrutura do Relatório

Capa

Imagen UBI/FE/DI

Licenciatura em...

UC: Bases de Dados

Título

Imagen ilustrativa/representativa do trabalho

Identificação dos elementos do grupo, ordenados por ordem crescente do número de estudante.

Local e data.

Agradecimentos

Incluir eventuais agradecimentos.

Resumo

Incluir um breve resumo (300 palavras).

Palavras-chave: 5 palavras ordenadas alfabeticamente.

Índice Geral

Lista de Abreviaturas

Lista de Fíguras

Lista de Tabelas

1. Introdução

1.1 Enquadramento

1.2 Motivação

1.3 Objectivos

1.4 Organização do documento

2. Desenvolvimento da aplicação cliente/servidor sobre bases de dados

2.1 Introdução

- 2.2 Aplicação cliente/servidor
- 2.3 SQL Server
- 2.4. Configuração do acesso ao servidor
- 2.5 *Frontend Python* (ou outro ambiente de desenvolvimento)

3. Modelação

- 3.1 Introdução (incluir a notação usada)
 - Fazer uma breve introdução à modelação de dados.
- 3.2 Descrição da organização
 - Indicar e justificar as opções tomadas para as situações não especificadas no enunciado.
- 3.3 Modelo de dados
 - Incluir imagens de progresso (ERD Plus).
 - Esquema relacional
- 3.3 Considerações
 - Eventuais considerações sobre o modelo final e sua implementação.

4. Aplicação

- 4.1 Distribuição de tarefas
 - Incluir uma lista com as tarefas (para a execução do trabalho prático) e quem ficou encarregue de as realizar.
 - 4.1.xx Descrição precisa das tarefas (cada elemento do grupo faz a sua) Cada elemento do grupo deve incluir uma subsecção com uma descrição muito precisa das suas tarefas. Pode incluir pseudo-código se tal for necessário, que pode detalhar e desenvolver nos apêndices.
- 4.2 Acesso à base de dados
 - Documentar, devidamente, a forma como foi efectuado o acesso à base de dados. Incluir excertos de código ilustrativos.
- 4.3 Funcionalidade
 - 4.3.1 Descrição geral
 - Descrição da funcionalidade global, incluindo uma representação esquemática de como funciona a solução.
 - 4.3.2 Aplicação
 - Descrição da funcionalidade sob os pontos de vista do utilizador (incluir imagens representativas – *screenshots*) e do programador (incluir excertos de código e SQL). Incluir os cenários descritos para cada aplicação.

5. Conclusões

- Indicar o que foi conseguido.
- Indicar o que não foi conseguido. Indicar a(s) razão(ões).

Epílogo

Incluir uma reflexão crítica sobre a disciplina (pontos a manter, a alterar e a eliminar).

Referências Bibliográficas

Usar uma norma para as referências.

Anexos

Colocar aqui o que for acessório para a leitura do trabalho (não desenvolvido pelos autores).

Apêndices

Scripts (criar bd, criar tabelas e restrições, dados iniciais).

Outros elementos desenvolvidos pelos autores, mas acessórios para a leitura do trabalho.

7. Notas finais

(A desenvolver em futuras versões)

- 5) Cada aluno pode ter uma classificação diferente.

Nas aulas práticas vai sendo avaliado o desempenho de cada aluno. Nesta avaliação pretende-se aferir a qualidade e a quantidade de trabalho realizado vs. previsto. A identificação das tarefas a desempenhar por cada aluno deve ser decidida dentro do grupo de trabalho. A estimativa dos prazos para a conclusão das tarefas é feita por cada aluno em colaboração com o grupo em que está inserido. Cada aluno deve realizar o seu conjunto de tarefas de modo a não perturbar o trabalho dos restantes elementos do grupo – e isto vai ser aferido (e classificado) pelos docentes.

- 6) As aplicações devem ser desenvolvidas em *Python*. Cada estudante deve desenvolver uma aplicação, devendo coordenar a sua atividade com os restantes membros da equipa.
- 7) Todos os estudantes têm de participar em todos os elementos de avaliação: relatório, aplicação, modelação e scripts.
- 8) Todos os estudantes têm de participar na discussão final do trabalho (trata-se de um dos critérios para obter a classificação de Frequência).