Curso Prático:

# INTRODUÇÃO À BASE DE DADOS

Utilizando Microsoft Access

Este texto destina-se a utilizadores que se iniciem na construção e utilização de pequenas base de dados pessoais.

José Coelho 2011

# Índice

# Índice de Figuras

# 1. INTRODUÇÃO

Uma **base de dados** é um local onde pode ser guardada informação. A informação pode ser consultada, alterada, apagada, na totalidade ou parcialmente, através de uma aplicação conhecida como Sistema de Gestão de Base de Dados (SGBD), também chamada simplesmente de Base de Dados (BD).

Ao contrário dos documentos de diversos tipos, em que a informação é colocada conforme o utilizador entender, numa base de dados a informação encontra-se estruturada, facilitando assim a utilidade e longevidade da informação, que de outra forma faria sentido para um utilizador num dado momento, e assim poderá ser útil para muitos utilizadores num período mais longo de tempo.

Uma base de dados pode ter diversos modelos que definem como a informação é organizada internamente. Os mais comuns são o **modelo hierárquico**, em que cada registo possui um e um só pai, tal como os ficheiros e pastas no computador, não podendo um registo possuir mais que um pai, o **modelo em rede**, que é idêntico ao modelo hierárquico mas cada registo pode possuir mais que um pai, e o **modelo relacional**, o mais comum e abordado neste texto.

No modelo relacional são definidas para cada tipo de entidade uma tabela, e para cada atributo de uma entidade uma coluna na tabela. As entidades em si são colocadas em linhas na tabela correspondente, com o valor de cada atributo na respectiva coluna.

Por exemplo, a base de dados do Moodle necessita de guardar informação sobre os utilizadores. Essa informação é guardada na tabela mdl_user, da qual se mostra uma parte na Figura 1-1.

| id | auth | confirmed | policyagreed | deleted | mnethostid | username |
|---|---|---|---|---|---|---|
| 1 | manual | 1 | 0 | 0 | 1 | guest |
| 2 | manual | 1 | 0 | 1 | 1 | jcoelho@univ-ab.pt.12069 |
| 3 | email | 1 | 0 | 1 | 1 | jcoelho72@gmail.com.12 |
| 4 | manual | 1 | 0 | 0 | 1 | vgama |
| 5 | manual | 1 | 0 | 1 | 1 | dcao |
| 6 | manual | 1 | 0 | 0 | 1 | jcesar |
| 7 | manual | 1 | 0 | 0 | 1 | heraclito |
| 8 | manual | 1 | 0 | 0 | 1 | ahenriques |
| 9 | manual | 1 | 0 | 1 | 1 | fernando |

Figura 1-1 – tabela mdl_user do Moodle

As colunas são atributos dos registos da entidade utilizador na aplicação Moodle. Nas linhas está informação sobre cada utilizador.

O leitor poderá estar a questionar-se nesta altura para que necessita de saber isto? Utiliza o Moodle e nunca viu esta tabela. Não será esta matéria relevante apenas para informáticos?

É verdade que os utilizadores interagem com as base de dados através de aplicações, como é o caso do Moodle, mas também há hoje em dia uma cada vez maior necessidade de base de dados locais, construídas e desenvolvidas para facilitar a resolução dos problemas de cada pessoa, ou de um

pequeno conjunto de pessoas. Tal como as folhas de calculo eram no passado utilizadas por um restrito número de pessoas enquanto que para outras era suficiente uma calculadora, e hoje em dia são uma ferramenta essencial em praticamente todas as actividades, com as base de dados está também a acontecer o mesmo. Guardar cada vez mais informação numa folha de calculo não é solução, apesar das folhas de calculo estarem bastante optimizadas, não permitem a flexibilidade na consulta de informação que uma base de dados permite.

Exemplos de informação a colocar numa base de dados pessoal: contactos de amigos; livros; CDs/DVDs; contas da casa. Numa empresa: clientes; empregados; ordenados; horários; equipamentos; projectos. As base de dados permitem a consulta/edição de vários utilizadores em simultâneo, no entanto este texto é dirigido ao leitor que pretende construir uma base de dados para uso próprio, e não pretenda ter de aprender uma linguagem de programação para construir a sua base de dados. A ferramenta ideal nestas condições é a utilização do *MS Access*.

O *MS Access* é uma base de dados que para além das operações normais de base de dados, de criação de tabelas e edição/consulta de dados, permite a criação de formulários e relatórios sem necessidade de programação, para que o utilizador aceda à base de dados não directamente a partir das tabelas, mas utilizando formulários para introdução e visualização de registos, e relatórios para consultas envolvendo vários registos. Dessa forma os dados podem ser verificados na introdução e corrigidos se não forem válidos, e assim manter a informação na base de dados coerente, de modo a permitir a qualquer momento obter extrair informação actualizada da base de dados.

Mesmo consultas complexas são possíveis no *MS Access* através de um "Wizard" em que a consulta é colocada de forma gráfica, evitando assim que o utilizador necessite de conhecer a linguagem SQL.

Embora seja desenhado para não necessitar programação e para um ambiente amador, o *MS Access* permite a utilização da linguagem SQL, pode ser utilizado numa aplicação web tal qual outra qualquer base de dados, e permite programação dentro dos formulários/relatórios. Não é no entanto aconselhada a sua utilização em sistemas de elevada utilização.

Neste texto não será dada relevância à interface do *MS Access*, mas será seguido um exemplo em que serão colocados imagens de várias fases de utilização da aplicação, na versão 2007. A evolução do *MS Access* é mais marcante a nível da interface, mantendo as mesmas funcionalidades ao longo do tempo. Mesmo o *MS Access 2.0*, do qual o autor teve acesso ao manual de utilizador impresso que data do ano 1994, inclui tudo o que mais importante existe na versão de 2007, e seleccionado para este texto, pelo que o leitor não terá problemas em seguir o exemplo com qualquer outra versão do *MS Access*. A generalidade das base de dados mantêm-se constante ao longo do tempo, o que é uma segurança acrescida para quem as utiliza.

A criação de uma base de dados local não pode ser confundida com o desenho de uma base de dados que sirva de base a uma ou mais aplicações informáticas, e utilizada por vários utilizadores em simultâneo. No primeiro caso pode ser feito por qualquer pessoa, as decisões tomadas apenas implicam a base de dados local. Já o segundo caso requer um profissional experiente. O desenho da base de dados irá influenciar não só o desenvolvimento das aplicações, como também a sua manutenção após estar em produção, e os próprios utilizadores das aplicações que utilizam a base de dados.

Utilizaremos o *MS Access*, e neste caso a criação de uma base de dados corresponde simplesmente à criação de um ficheiro do *MS Access* no disco rígido, como se pode ver na Figura 1-2.

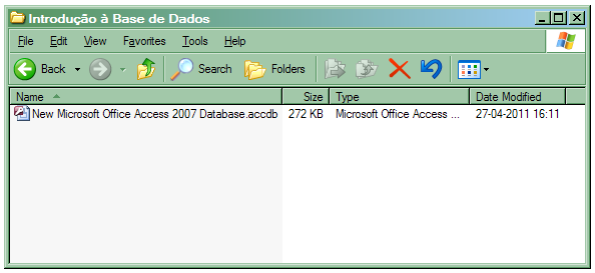


Figura 1-2 – Criação de uma base de dados

Convém realçar no entanto que um ficheiro de *MS Access* não é como outro ficheiro de uma aplicação de folha de calculo ou processamento de texto. Nestas aplicações edita-se um documento, e no final pode-se gravar ou não. Um ficheiro *MS Access* está sempre gravado. Logo que se altere algo na base de dados, essa alteração é imediatamente reflectida, pelo que não existe o conceito de gravar.

Numa base de dados pode-se e deve-se fazer cópias de segurança (backups), que no caso do *MS Access* basta que se duplique o ficheiro em disco. Caso seja feito algum erro e existam dados perdidos, pode-se recuperar os dados parcialmente da cópia de segurança, ou simplesmente substituir a base de dados pela cópia de segurança mais recente, perdendo-se no entanto as alterações desde que a cópia de segurança foi criada.

Para base de dados profissionais, tanto a criação como as cópias de segurança não são feitas desta forma, mas são em tudo operações semelhantes. As cópias de segurança são feitas automaticamente normalmente com uma frequência diária.

Abrindo o ficheiro criado, ainda completamente vazio, obtém-se o *MS Access* pronto a aceitar informações, como indicado na Figura 1-3.

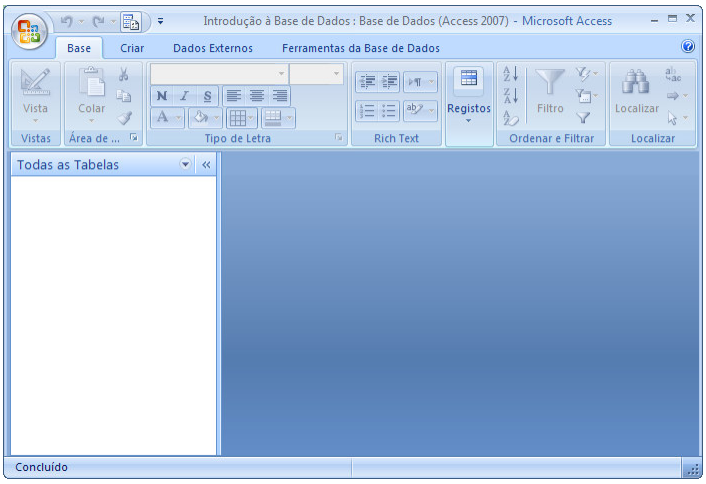


Figura 1-3 – Ficheiro do MS Access vazio

O *MS Access* tem os menus normais presentes nas aplicações Office, no friso Base, e mais três zonas: criação da elementos na base de dados (Criar); importação de dados (Dados Externos); operações sobre a Base de Dados (Ferramentas da Base de Dados). Na Figura 1-4 estão os 3 frisos referidos.


Figura 1-4 – Frisos do MS Access

Na zona da criação, pode-se criar novas tabelas, formulários, relatórios, consultas e macros (friso Outro). Neste texto não nos interessa as macros, porque envolvem programação. De resto, podemos criar tabelas para guardar informação, criar formulários para introduzir informação, e relatórios para ver informação. Com as consultas podemos extrair informação precisa, ou suportar a construção de relatórios e formulários mais evoluídos.

Na zona de importação de dados pode-se importar/exportar informação de/para diversos formatos. Estas operações devem ser feitas com a máxima atenção, dado que normalmente envolvem muitos registos e algum erro pode ser replicado em muitos lados, o que pode por vezes ser difícil a sua correcção. É essencial a qualquer base de dados que tenha forma de introduzir e extrair informação em grandes quantidades, mas não é essencial para o objectivo deste texto.

Na zona das operações sobre a base de dados pode-se efectuar análises à base de dados, bem como editar relações entre tabelas, e gerir ligações externas a outras base de dados. Este friso é ainda menos relevante que o anterior para os objectivos deste texto, exceptuando-se as relações entre tabelas que permite visualizar e editar a estrutura global da base de dados.

# 2. TABELAS

O primeiro passo na criação de uma base de dados é definir os tipos de entidades necessárias na base de dados, e respectivos atributos. Aos tipos de entidades correspondem tabelas, e aos atributos as colunas.

Este é o primeiro passo porque as tabelas a criar não dependem de nada, apenas das nossas necessidades de representação. Praticamente tudo o resto na base de dados depende das tabelas existentes. Deste passo deve resultar no conjunto final de tabelas, eventualmente sem os atributos na sua forma final, de forma a evitar ter de rever todos os passos seguintes.

Para que este passo seja realizado o mais correctamente possível, é necessário que se tenha bem claro qual o objectivo e o que se pretende com a base de dados, basicamente para poder responder à questão:

**Quais são as entidades necessárias no nosso sistema?**

Com a resposta a esta questão constroem-se as tabelas, agrupando entidades do mesmo tipo, mas tendo em atenção ao seguinte:

- Uma e uma só tabela por cada tipo de entidade relevante;
- Uma e uma só coluna para cada atributo de um tipo de entidade.

Os princípios parecem bastante claros, mas na prática cometem-se bastantes erros. Realçamos três erros principais.

### **Erro 1:** Juntar alhos com bugalhos

Naturalmente que se duas entidades do mesmo tipo, têm atributos distintos, deve-se suspeitar se essas entidades devem estar na mesma tabela, ou se por outro lado se deve criar tabelas distintas. Uma tabela tem um conjunto de colunas fixo, e se bem que para alguns registos umas colunas possam ficar vazias e outras preenchidas, o ideal é que quando uma linha seja preenchida, seja preenchida completamente, não fiquem buracos vazios. Para realçar essa necessidade, no caso limite pode existir apenas uma só tabela, com todas as colunas existentes em todas as entidades. Neste caso seria possível introduzir toda a informação pretendida, mas perde-se a noção de estrutura, e dificultava-se a construção de consultas e utilização dos dados, para não falar do possível desperdício de espaço em disco.

**Exemplo:** Um exemplo de um erro deste tipo é a criação de uma tabela de "Artigos" e nessa tabela tanto colocar aparelhos eléctricos em que um dos atributos é a potência, que não faz sentido em outros artigos não eléctricos, por exemplo uma lata de tinta, cujo atributo como a referência à cor, ao volume, ou às características da tinta, não tem sentido para artigos eléctricos.

**Resolução:** Caso se pretenda o conceito de artigo, deve-se manter uma tabela de artigos em que tem o nome do artigo e o tipo de artigo e outras colunas que sejam comuns a todos os artigos. Deve-se criar uma tabela para cada tipo de artigo, com as colunas que fazem sentido para esse tipo de artigo.

### **Erro 2:** Dividir para complicar

Se duas ou mais tabelas têm o mesmo conjunto de colunas, há que desconfiar se são realmente entidades distintas, ou se são a mesma entidade mas com categorias distintas. Nesse caso pode-se criar uma só tabela, em que para além das colunas já existentes, cria-se uma coluna com o nome da categoria a que o registo pertence.

**Exemplo:** Por exemplo, foi criada uma tabela por cada distrito, com informações dos concelhos do distrito. Caso seja criado um novo distrito, muito embora tal não seja vulgar, teria de ser criada nova tabela. Ora as tabelas serem imutáveis com o tempo. Esta solução torna o número de tabelas dependente do número de dados (distritos), e dificulta consultas que envolvam mais que um distrito.

**Resolução:** Criar uma tabela concelhos, com a junção das tabelas dos distritos e adicionando uma coluna extra "Distrito" com o nome do distrito a que o concelho pertence.

### **Erro 3:** Caldeirada

Se uma coluna tiver uma lista de valores, então há provavelmente uma tabela a menos. Nesse caso pode-se apagar a coluna e criar uma tabela para a coluna, com cada valor numa linha distinta.

**Exemplo:** Por exemplo, um registo de um aluno numa cadeira tem um campo de avaliação contínua com as notas ao longo do semestre, separadas por vírgulas.

**Resolução:** Criação de uma tabela de "Trabalhos", com cada nota do aluno para cada trabalho, mantendo naturalmente a referência ao aluno e à cadeira a que a nota diz respeito.

Vamos ilustrar a criação de uma base de dados com o exemplo de uma pequena biblioteca, em que se pretende: **registar as requisições de livros, bem como os estados das cópias de cada livro.**

Comecemos por uma primeira tabela, de Livros.

A criação de uma tabela de Access basta carregar no Criar/Tabela e avança-se para a edição de uma tabela em Vista de Estrutura, em que se pode especificar as colunas e o nome da tabela, bem como os tipos de cada coluna, como se pode ver na Figura 2-1.

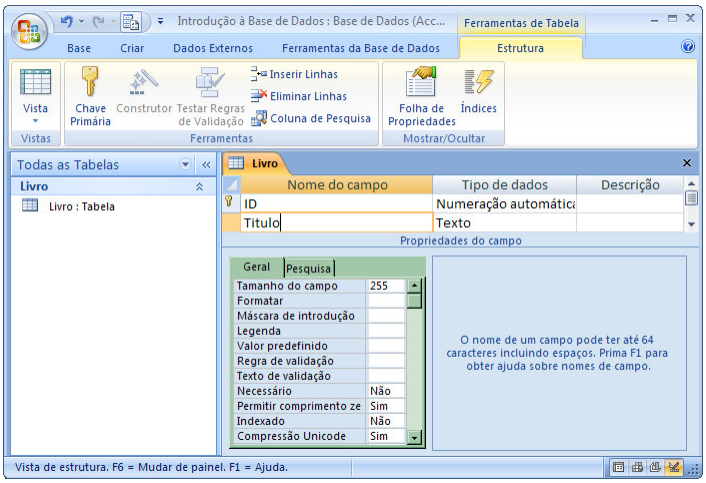


Figura 2-1 – Criação de uma tabela

Notar que esta tabela após criada, contem já uma coluna, a coluna ID, do tipo "Numeração automática". Essa coluna é colocada de omissão e o seu valor é um número inteiro automático e crescente, não existindo duas linhas com o mesmo número nessa tabela. Tal é útil para identificar cada linha da tabela univocamente, e que será utilizado para definir relações entre entidades. Alternativamente pode-se definir um conjunto de colunas como sendo as colunas chave, e dessa forma não haverá na tabela duas linhas exactamente com os mesmos valores nas colunas chave. Aconselha-se a manter as colunas ID para identificação das linhas na tabela.

Cada campo pode ter um conjunto de características definidas, para além do tipo. Estas características podem servir para refinar uma base de dados em utilização, principalmente se for utilizada por várias pessoas, convém colocar por exemplo uma descrição em cada campo, regras de validação, entre outras, mas sem grande relevo para os objectivos deste texto. Terá hipótese de explorar estas possibilidades, e as características e farão mais sentido, após ter construído algumas base de dados. Para já interessa distinguir os campos do tipo texto dos numéricos, e eventualmente o tipo data.

Acrescentou-se na tabela Livro, para além do título, os campos autor, editora, ISBN e ano. Apenas o campo ano foi colocado como numérico, sendo os restantes deixados como texto. Após a definição da tabela, obtém-se a seguinte tabela no modo Vista de Folha de Dados, pronta a receber dados, na Figura 2-2.

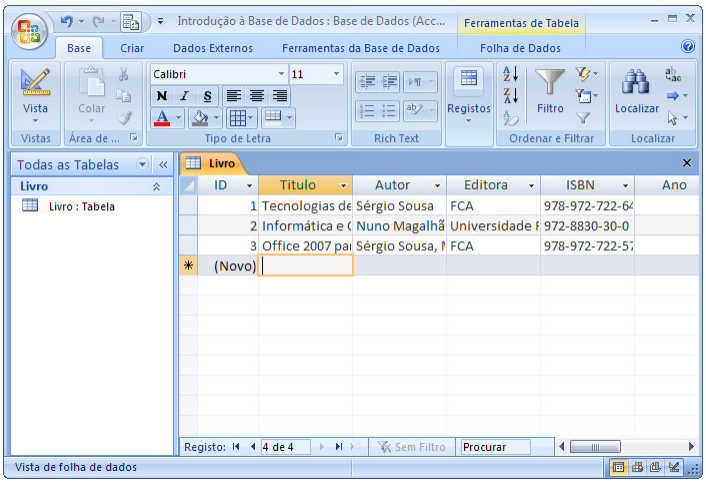


Figura 2-2 – Tabela Livro após criada e introduzidos alguns dados

Para além dos livros, que outras tabelas necessita a base de dados da biblioteca do nosso exemplo? Provavelmente o leitor chegou também a estas tabelas:

- Leitor;
- Requisição;
- Cópia Livro.

A entidade "leitor" não foi referida propositadamente no pequeno enunciado que descreve esta base de dados, pelo que poderia não ter ser criada, e também provavelmente a entidade Cópia Livro não teria sido criada agora não fosse a expressa referência no enunciado. A parte do trabalho mais importante é a identificação do que é e não é relevante para a base de dados em questão. Por exemplo, uma tabela com o funcionário poderá não ser relevante, já que a biblioteca tem provavelmente dois ou três funcionários, o que não seria o caso de uma grande biblioteca.

Caso não tivesse sido criada a tabela Leitor seria tentado a criar um atributo na tabela Livro, com o nome dos leitores que têm requisições do livro, ou uma lista com as cópias do livro. Essa situação deve ser evitada para não se cair no erro anteriormente identificado como 3, ou seja, num atributo colocar uma lista de valores, e não apenas um só valor.

O leitor pode acompanhar este exemplo, criando as tabelas referidas, introduzindo alguns dados e colunas que achar conveniente. É natural que as colunas, e nesta fase, mesmo as tabelas, possam não ter a sua utilidade clara, e portanto podem ser alteradas. No entanto, quando se começar a introduzir dados em quantidade numa base de dados, alterações a este nível tornam-se cada vez mais complicadas.

As tabelas criadas não são completamente independentes. A existência de uma entidade pode requerer a existência de uma ou mais outras entidades, pertencentes a outras tabelas. No exemplo dado, uma cópia de um livro apenas faz sentido se existir o registo do próprio livro. Uma requisição apenas faz sentido se existir um leitor e uma cópia do livro.

As dependências entre tabelas não é um problema, aliás, devem existir. Se não existirem não faz sentido numa mesma base de dados estarem tabelas que nada têm a ver uma com a outra. Mais vale

nesse caso criar duas base de dados distintas, e serem mantidas e actualizadas de forma independente.

Pode-se especificar estas restrições na própria base de dados, para que seja garantido que a informação inserida é coerente (Relações). É isso que é feito na Figura 2-3 para as tabelas criadas.

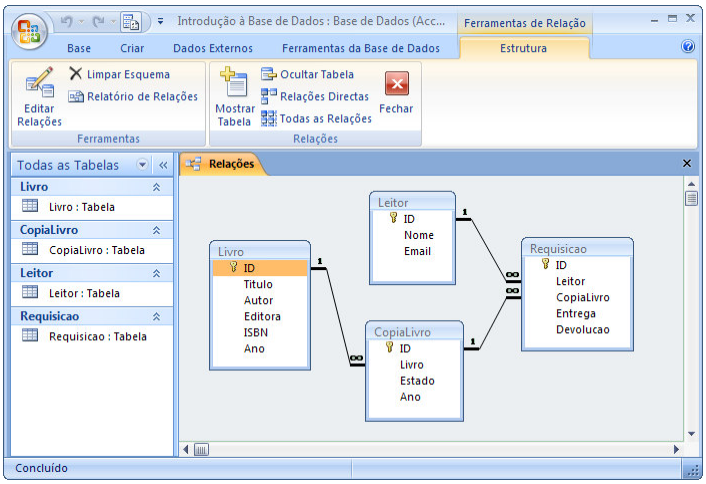


Figura 2-3 – Relações entre tabelas

No diagrama de relações, cada elemento é uma tabela, tendo a lista de colunas disponíveis. Todas as tabelas criadas têm a coluna ID para identificação de um elemento na tabela, e outras colunas contendo informação do elemento. Algumas destas colunas são referências a elementos de outras tabelas. Uma CopiaLivro tem um campo Livro, cujo valor é o número identificado o livro que é realmente copiado. Como campos específicos tem o estado da cópia do livro, e o ano de aquisição da cópia. Por sua vez, o leitor tem como campos o nome e email. A tabela Requisição tem dois campos, um para a cópia do livro requisitado, e outro para o leitor que o requisita, ambos os campos referências a elementos de outras tabelas. Como campos próprios tem apenas a data de entrega e de devolução do livro. No final, todas as tabelas encontram-se ligadas por relações.

Para colocar uma relação entre tabelas, basta que arraste de um campo para o outro. Se não criou os campos de ligação entre tabelas, crie, e coloque estas restrições entre tabelas[1].

Após colocar relações entre tabelas, a base de dados não irá aceitar uma requisição sem que exista um leitor, ou uma cópia de um livro. Por outro lado, não é possível inserir uma cópia de um livro, sem que exista um livro. Desta forma fica garantida a coerência da informação, algo que não aconteceria se tivéssemos a utilizar uma folha de calculo para guardar esta mesma informação.

Alternativamente à colocação de relações entre tabelas, pode-se garantir na aplicação de interface com a base de dados, seja via web, seja via um formulário no *MS Access*, que a integridade da informação é mantida, requerendo no entanto programação. É no entanto aconselhada a primeira hipótese, colocar as restrições a nível da base de dados, uma vez que a aplicação de interface pode ser complexa e poderá não ser simples manter a integridade dos dados, e no caso de existir algum bug na aplicação, os dados poderiam ficar comprometidos.

[1] Tem de seleccionar a caixa de selecção “Impor integridade referencial” para que seja mantida a coerência da base de dados.

As relações inseridas são do tipo **1-para-N**, isto é, um Livro pode estar associado a N cópias, sendo N um número natural qualquer. No lado do 1 (Livro) está a chave, neste caso a coluna ID, que corresponde ao identificador único, e no lado do N (CopiaLivro) está um campo com o nome da tabela (Livro) e do tipo numérico, que deverá ter um valor existente na coluna ID da tabela Livro.

Existem também relações do tipo **1-para-1**, ou seja, a cada registo da tabela A está associado um só registo da tabela B. Este tipo de relações não são vulgares, e possivelmente resultam de divisões de uma tabela. Um bom motivo para essa divisão é o exemplo dos Artigos, em que alguns são electrodomésticos, outros são latas de tinta. Neste caso faz sentido existir um registo na tabela de artigos por cada registo de um electrodoméstico e lata de tinta, devendo os registos terem uma relação de 1-para-1. Um mau exemplo é a divisão de uma tabela, com exactamente o mesmo número de registos, em duas tabelas, permanecendo os registos ligados por uma relação 1-para-1. Estas relações são implementadas na BD como se fossem uma relação 1-para-N, através da colocação de uma coluna numa das tabelas (ou em ambas), para o registo correspondente na outra tabela.

O último tipo de relações é o **N-para-N**, em que elementos de ambas as tabelas podem estar associados sem qualquer restrição. Um exemplo dessa relação é entre uma tabela de alunos e disciplinas. Um aluno pode estar inscrito a mais que uma disciplina, e cada disciplina pode ter mais que um aluno inscrito. Estas relações são implementadas com recurso a uma tabela extra, que neste caso tem um nome claro, a tabela de inscrições, com uma relação de 1-para-N da nova tabela para as restantes duas. Na tabela de inscrições coloca-se um campo com a identificação do aluno, e outro com a identificação da disciplina, portanto duas relações 1-para-N. No exemplo da biblioteca, a tabela Requisicao pode ser vista como a implementação de uma relação N-para-N entre os leitores e as cópias de livros.

A informação das relações entre tabelas permite não só assegurar que a informação se mantém coerente, como permite outro tipo de navegação nas próprias tabelas.

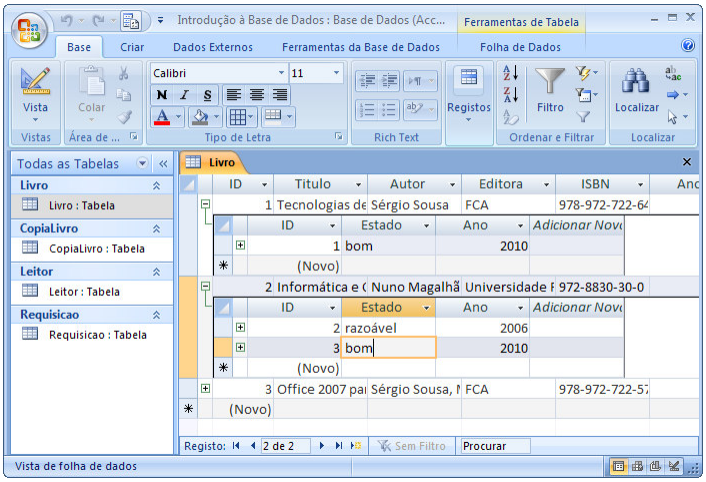


Figura 2-4 – Navegação pela informação, devido às relações entre tabelas estarem definidas

O livro está ligado a cópias do livro, pelo que em cada linha da tabela do livro, passa agora a ser possível ver quais as cópias que estão associadas ao livro. Da mesma forma é possível ver em cada cópia do livro a lista de requisições em que a cópia do livro foi utilizada.

A visualização de uma tabela oferece desde logo uma forma rápida para introdução de dados. No entanto apenas dados experimentais devem ser colocados desta forma, sendo necessário para o dia-a-dia a criação de formulários de introdução de dados de modo a poupar o utilizador das tabelas necessárias, evitar tentações do utilizador em alterar a estrutura de dados por no momento não compreender a utilidade de uma tabela ou campo, e pedir-lhe apenas a informação necessária da forma mais simples possível.

# 3. CONSULTAS

As consultas é o que dá sentido à base de dados. Não vale a pena guardar informação se não se consultar mais tarde. Dependem directamente das tabelas e colunas criadas, ou seja, da estrutura da base de dados. Se a estrutura for boa, as consultas são fáceis de se realizarem. Caso a estrutura tenha sido mal pensada, as consultas são consideravelmente mais difíceis de se realizar, podendo por vezes ficar inviabilizadas em base de dados grandes, devido ao volume dos dados associado à falta de estrutura.

As consultas são feitas tanto pelas aplicações de interface da base de dados com o utilizador, ficando no caso do Moodle no código php, como pelo próprio utilizador que aceda directamente à base de dados e necessita de uma informação específica.

No *MS Access*, as consultas são úteis também para servir de base à construção de formulários e relatórios, que serão mais utilizados pelo utilizador, podendo no entanto o utilizador efectuar qualquer consulta directamente. A linguagem de consultas válida para todas as base de dados é o SQL, mas no *MS Access* existe um Assistente de Consultas que permite ao utilizador efectuar consultas sem ter de aprender essa linguagem.

O Assistente de Consulta é composto por duas partes. A primeira (Criar/Assistente de Consultas), inicia uma sequência de caixas de diálogo que após preenchidas é construída a consulta pretendida. A segunda parte permite editar através de caixas de diálogo uma consulta existente, ou criar uma nova (Criar/Estrutura da Consulta). Vamos dar relevo apenas à segunda parte, dado que se encontra mais perto da linguagem SQL, mas o leitor é livre de explorar o Assistente de Consulta.

Para criar uma consulta pode-se executar o comando (Criar/Estrutura da Consulta), obtendo-se uma caixa de diálogo que nos permite seleccionar as tabelas ou outras consultas a mostrar, como mostra a Figura 3-1.

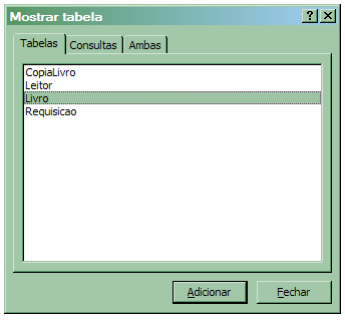


Figura 3-1 – Seleccionar as tabelas que estão envolvidas na consulta

Começamos primeiramente por um exemplo de uma consulta simples, pretendemos ver todos os **livros com edição superior ao ano 2000**. Esta consulta só envolve uma tabela, a tabela Livros, pelo que selecciona-se a tabela e fecha-se a caixa de diálogo. Entra-se de imediato na edição da consulta, podendo-se chamar a caixa de diálogo anterior a qualquer altura através do botão "Mostrar Tabela", que se pode ver na Figura 3-2.

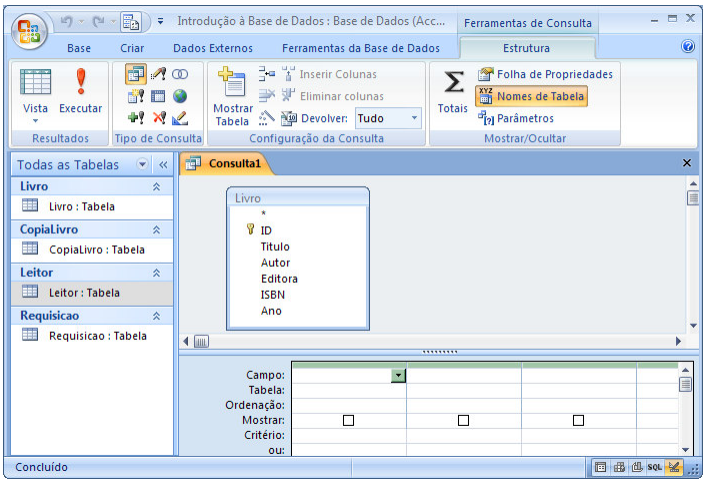


Figura 3-2 – Edição de uma consulta à tabela Livro

A consulta encontra-se dividida em duas zonas. A zona de cima tem a parte da estrutura da base de dados que se pretende consultar, neste caso apenas a tabela Livro. Na zona de baixo coloca-se a informação sobre a consulta. Em cada coluna pode-se colocar (por selecção ou arrastando uma coluna de cima), uma coluna das tabelas disponíveis.

Podemos começar por mostrar o título, o autor e a editora. Automaticamente os campos ficam seleccionados a "Mostrar", o que quer dizer que essas colunas vão ser visualizadas no resultado da consulta. As linhas de "Ordenação" e "Critério" permanecem em branco. No entanto convém que o resultado da consulta seja ordenado, de forma a poder-se encontrar um livro facilmente, pelo que pode-se alterar o valor da coluna do "Titulo" na parte de ordenação para "Ascendente".

Para podermos colocar a nossa restrição, livros com edição superior a 2000, temos de colocar também a coluna "Ano" mesmo que não a mostremos na consulta, e colocar a restrição em "Critério" que tem que ser superior a 2000. Podemos ver o resultado na Figura 3-3.

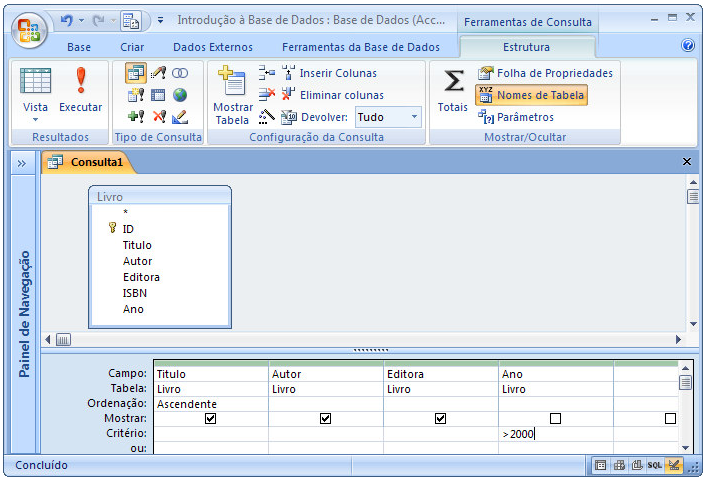


Figura 3-3 – Consulta que mostra os livros com ano de edição superior a 2000

Pode-se agora gravar a consulta, para "LivrosRecentes", a qual ficará disponível para ser chamada a qualquer altura, em qualquer local. Na Figura 3-4 pode-se ver a consulta junto da tabela Livros, dado que utiliza apenas esta tabela, e o resultado da execução da consulta.

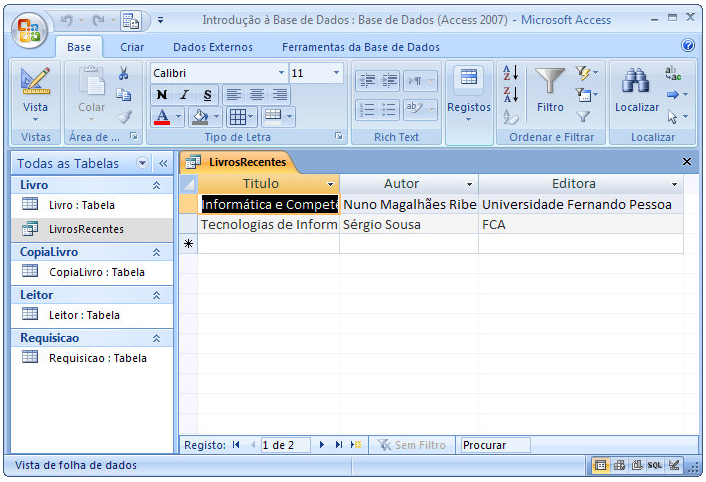


Figura 3-4 – Resultado da consulta definida na Figura 3-3

Embora a complexidade das consultas dadas neste texto é suficiente utilizar o Assistente de Consultas, a linguagem SQL permite também especificar a mesma consulta e apresenta um formato simples, sendo interessante a sua introdução para por um lado melhor cimentar a matéria, e por outro servir como alternativa para alguns estudantes. A correspondência em SQL da vista de estrutura, pode-se ver e editar trocado a vista para "Vista de SQL". Vejamos como fica a consulta acima em linguagem SQL:

```
SELECT Titulo, Autor, Editora
FROM Livro
WHERE Ano>2000
ORDER BY Titulo;
```

A negrito estão destacadas as palavras que pretendem à linguagem SQL, o resto faz parte da base de dados. A instrução SELECT selecciona um conjunto de campos (neste caso 3), da tabela Livro, sendo utilizada a instrução FROM para indicar a origem dos dados, que pode ser uma tabela ou consulta existente, seguida da instrução WHERE onde se colocam as restrições a executar, e finalmente a instrução ORDER BY especifica a coluna pela qual as linhas devem ser ordenadas. As instruções WHERE e ORDER BY são opcionais, e se não existirem são mostradas todas as linhas, pela ordem que estão na tabela. Ao contrário dos textos na aplicação, que podem variar conforme a linguagem activa, os nomes das instruções na linguagem SQL não muda, sendo válidos para qualquer base de dados.

De uma forma geral, pode-se utilizar a linguagem SQL com a seguinte forma:

```
SELECT lista de campos a mostrar
FROM origem da informação a mostrar
WHERE condições a satisfazer nas linhas a mostrar
ORDER BY campos pelo qual se deve ordenar o resultado
```

Estas consultas acrescentam alguma flexibilidade ao equivalente no Excel, em que se pode por exemplo utilizar filtros, mas apenas envolvem uma tabela, e pouco mais. Até aqui foi possível

seleccionar as colunas a visualizar, ordenar por uma coluna, e restringir as linhas a mostrar por uma condição que envolva uma coluna, mesmo que a mesma não seja mostrada.

Suponhamos agora que queremos efectuar uma consulta com várias tabelas. Pretende-se ter a lista de **livros lidos por cada leitor, independentemente da cópia do livro utilizada**. Nesta consulta estão envolvidas todas as tabelas, pelo que tem que se adicionar todas as tabelas. No entanto pretende-se apenas a lista dos leitores e os títulos dos livros lidos, pelo que coloca-se na consulta apenas essas colunas. Para facilitar a leitura da consulta, é definida a ordem primeiro por nome do leitor e depois por título do livro.

Pretende-se ainda alterar o nome das colunas, o que é normal em consultas com mais de uma tabela, dado que o nome das colunas faz sentido apenas no contexto da tabela. Se não se alterar nada ficariam as colunas "Nome" e "Título", em vez disso pretende-se as colunas "Leitor" e "Livro". Pode-se alterar o nome colocando o nome de acordo com o realizado na Figura 3-5.

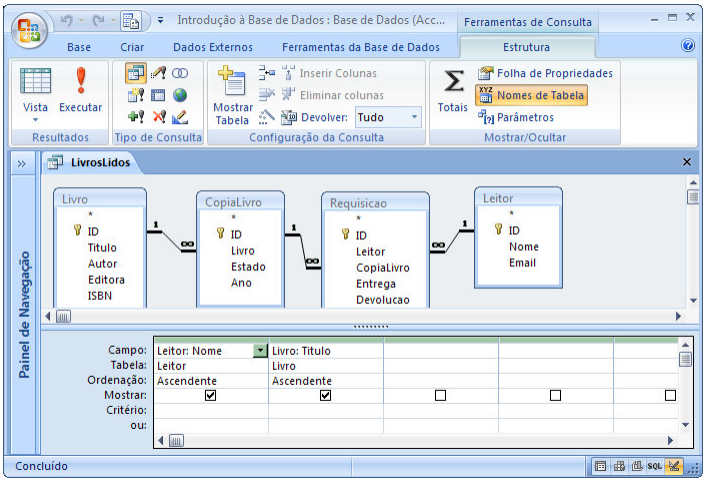


Figura 3-5 – Consulta com os livros lidos por cada leitor.

O resultado é o visualizado na Figura 3-6. Notar que a consulta LivrosLidos fica associada a todas as tabelas, porque utiliza realmente todas as tabelas, embora apenas sejam visualizadas duas colunas de duas tabelas.

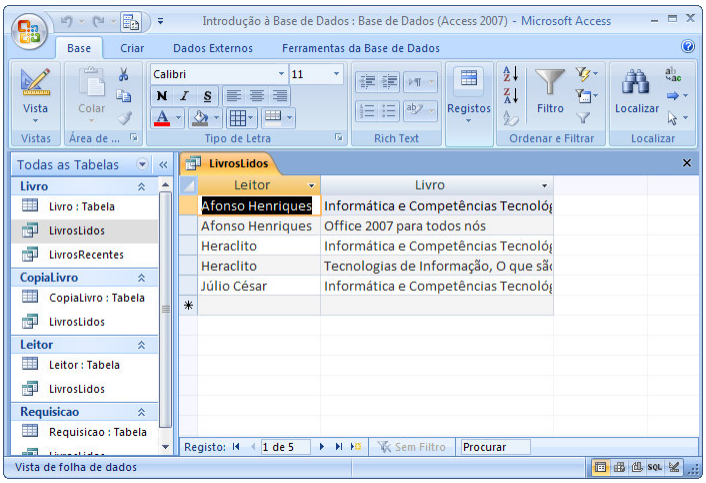


Figura 3-6 – Resultado da consulta da Figura 3-5

Neste caso o equivalente da consulta em linguagem SQL não utiliza uma tabela mas sim 4 tabelas, pelo que é um pouco mais complexa[2]:

```
SELECT Leitor.Nome As Leitor, Livro.Titulo As Livro
FROM Livro, CopiaLivro, Requisicao, Leitor
WHERE Livro.ID=CopiaLivro.Livro AND
      CopiaLivro.ID=Requisicao.CopiaLivro AND
      Leitor.ID=Requisicao.Leitor
ORDER BY Leitor, Livro;
```

Na alteração do nome de um campo é utilizada a palavra "As", ficando o nome anterior, identificado por "Leitor.Nome" para "Leitor". Os dois nomes a identificar um campo é útil quando se utilizam várias tabelas, já que podem haver campos com o mesmo nome mas a pertencer a tabelas distintas. Desta forma não há ambiguidades, sabe-se a tabela e o campo que se pretende identificar. O *MS Access* coloca sempre nas consultas que faz o nome da tabela e nome do campo, mas caso a consulta seja escrita manualmente não é necessário utilizar o nome da tabela se não existirem ambiguidades.

Para a origem dos dados, neste caso teve-se que colocar uma lista de tabelas: as 4 tabelas envolvidas. No Assistente de Consultas as relações entre tabelas globais são logo especificadas, mas na linguagem SQL tem de se especificá-las, pelo que no campo dos condicionais, tem que se colocar todas as restrições que fazem sentido utilizar estas tabelas em conjunto. Ou seja, o ID do livro tem de ser igual ao campo Livro da tabela CopiaLivro, tal como o ID da tabela CopiaLivro tem de ser igual ao campo CopiaLivro da tabela Requisição e o mesmo para o campo ID da tabela Leitor e o campo Leitor da tabela Requisicao. Os vários condicionais são juntos com a palavra "AND", de modo a que sejam todos satisfeitos. Finalmente ordena-se pelos dois campos existentes.

Até agora só se retornou um subconjunto das linhas e das colunas disponíveis. Pretende-se agora fazer consultas que agreguem informação de várias linhas. Por exemplo, pretende-se saber para cada livro, quantas cópias existem. Neste caso tem que se adicionar novamente as tabelas envolvidas, as tabelas Livro e CopiaLivro, adicionar as colunas que se pretenderem para identificar

[2] A junção de duas tabelas também se pode fazer através do INNER JOIN, em que se tem de identificar forçosamente a igualdade de dois campos da tabela, sendo esta a instrução utilizada pelo Assistente de Consultas. A utilização da lista de tabelas e dos condicionais no WHERE foi utilizada no texto por ser de leitura mais simples.

o livro. No entanto, como se pretende agrupar, tem que se carregar no botão "Totais " para visualizar-se uma linha na consulta de "Total". Nessa linha, pelo menos uma coluna na consulta deve estar no modo "Agrupar por", de forma a indicar como os registos devem ser agrupados.

Esta consulta é em tudo igual às consultas anteriores, mas as os registos que tiverem o mesmo valor na coluna em "Agrupar por" são juntos numa só linha. Por esse motivo, em todas as restantes colunas da consulta tem que se indicar uma função agregadora (soma, máximo, primeiro, contar, etc).

Foram adicionados dois campos para serem calculados. O primeiro é o campo ID da tabela CopiaLivro, com a contagem de registos. O segundo é o campo Ano da tabela CopiaLivro, em que é dada a média dos anos das cópias dos livros. Pode-se ver esta consulta na Figura 3-7.

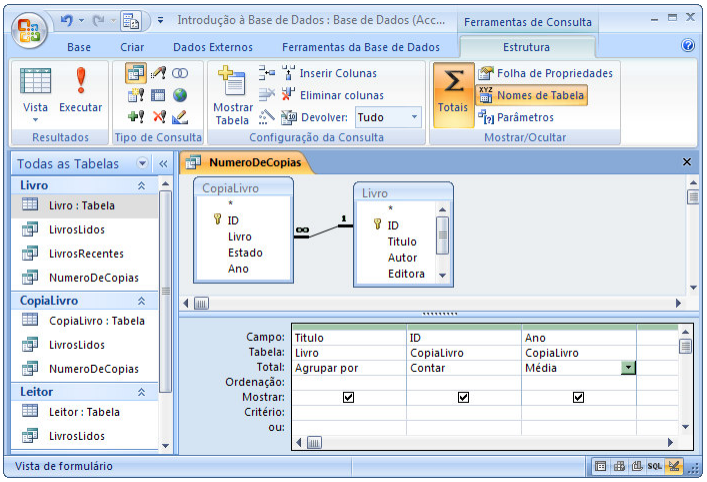


Figura 3-7 – Consulta agregando dados

O resultado da consulta é o seguinte pode-se ver na Figura 3-8.

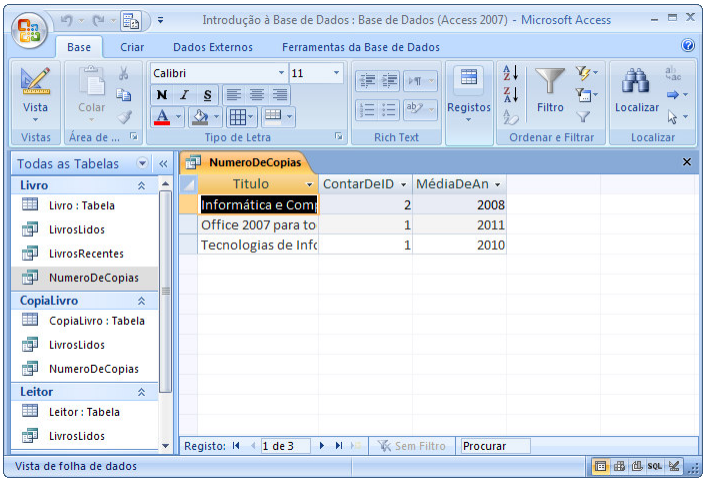


Figura 3-8 – Resultado da consulta da Figura 3-7

Este tipo de consultas é de elevada importância, dado que à medida que o volume de informação na base de dados aumenta, é necessário agregar informação de forma a visualizar-se informação útil. Uma tabela enorme de nada serve, e alguns casos particulares não são representativos de toda a

informação. A agregação da informação permite extrair informação útil de grande volume de dados, que neste caso poderia ser por exemplo a identificação dos livros com as cópias mais antigas.

Os condicionais continuam a poder serem colocados, antes e após a agregação. Por exemplo, pretende-se excluir as cópias em estado distinto de "bom", e pretende-se ver apenas os livros cujas cópias têm um ano de aquisição médio inferior ao ano 2010. Pode-se modificar a consulta anteriormente feita para reflectir esta situação, adicionando-se a coluna do estado, mas na linha do total colocar o "Onde ", dado que essa coluna não será agregada mas sim utilizada para filtrar antes da agregação, enquanto que na coluna do Ano coloca-se o condicional normalmente, de forma a ser realizado após a agregação sobre o valor agregado, isto é, a média do ano, como se pode ver na Figura 3-9.

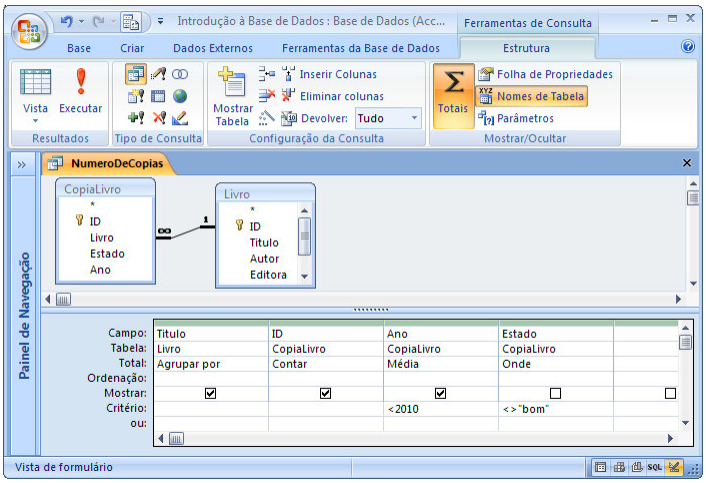


Figura 3-9 – Restrições em consultas com dados agregados

O resultado da consulta é mostrado na Figura 3-10.

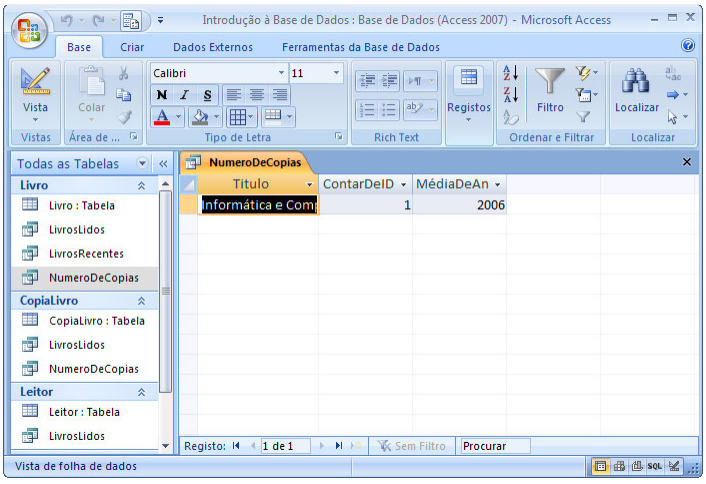


Figura 3-10 – Resultado da consulta definida na Figura 3-9

Esta consulta em linguagem SQL também segue o formato geral enunciado, mas tem de utilizar mais instruções, GROUP BY e HAVING:

```
SELECT Titulo, Count(CopiaLivro.ID) As "Nº Cópias", Avg(CopiaLivro.Ano) As
"Ano"
FROM Livro, CopiaLivro
WHERE Livro.ID=CopiaLivro.Livro AND
      Estado<>"bom"
GROUP BY Titulo
HAVING Avg(CopiaLivro.Ano><2010;
```

Notar que neste caso utilizou-se o campo "Titulo" sem a tabela correspondente numa consulta com várias tabelas. Isto é possível porque a única tabela com um campo "Titulo" é a tabela Livro. Para todos os campos a mostrar na consulta, que não sejam o campo que se está a agrupar, é necessário fornecer uma função para agregar os diversos resultados, sendo portanto conveniente alterar o nome de modo a não conter simplesmente o nome da função agregadora e o campo utilizado. Realçar também a utilização das aspas sempre que existem espaços ou valores constantes, como o caso de "bom" que desta forma não é confundido com o nome de um campo.

O formato mais genérico da linguagem SQL abordado neste texto, e também em termos de utilização do Assistente de Consultas, é o seguinte:

```
SELECT lista de campos a mostrar
FROM origem da informação a mostrar
WHERE condições a satisfazer nas linhas a considerar
GROUP BY campo ou lista de campos a agrupar
HAVING restrições nos campos agrupados
ORDER BY campos pelo qual se deve ordenar o resultado;
```

As consultas podem ir sendo refinadas de forma a ficarem cada vez mais próximas do que é pretendido, como aconteceu neste caso. Não há grande problema em alterar uma consulta, a não ser que a consulta seja utilizada por outras consultas, ou por um formulário ou relatório. Deve portanto nesta fase serem construídas todas as consultas necessárias, e no caso de serem necessárias mais consultas no futuro, pode-se sempre adicionar mais consultas em vez de editar consultas existentes e em utilização.

Este facto faz com que esta fase seja menos importante do que a anterior, porque pode-se sempre adicionar mais consultas. Já as tabelas não convém estar a adicionar e remover tabelas a qualquer altura.

# 4. FORMULÁRIOS

Para interagir com a base de dados pode-se utilizar no *MS Access* o acesso às tabelas e consultas directamente, de acordo com os capítulos anteriores. No entanto por vezes tem que se editar informação em várias tabelas, e poderá não ser esta a interface ideal, principalmente porque os utilizadores da base de dados podem não entender bem a estrutura existente, ou editarem partes que não devem editar no dia-a-dia.

No *MS Access* pode-se construir facilmente formulários e relatórios para facilitar a interacção com a base de dados, não excluindo a edição das tabelas e consultas directamente, mas deixando essa via fora das operações de rotina do dia-a-dia, em que facilmente se cometem erros e distracções.

Começando pelos formulários, tipicamente pode-se querer criar um formulário por cada tabela de informação. Pode haver no entanto tabelas para o qual tal não faça sentido. No nosso exemplo, é suficiente formulários para as tabelas Livro e Leitor. Seleccionando a tabela Livro, pode-se executar o comando Criar/Formulário e um formulário automático é desde logo criado sem ser pedida qualquer informação, como se pode ver na Figura 4-1.

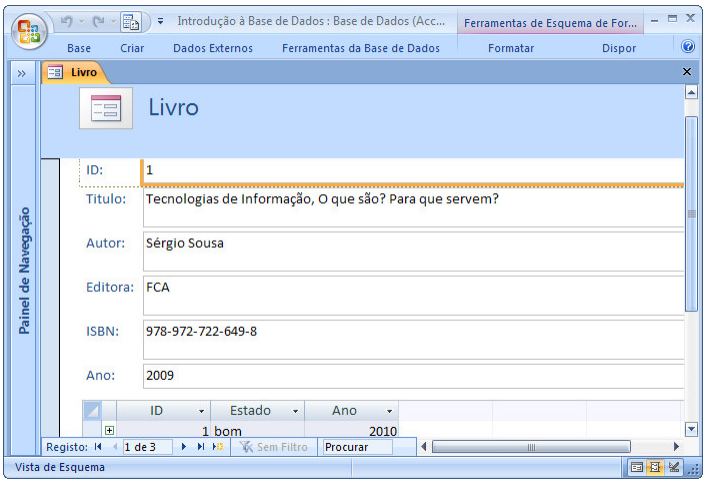


Figura 4-1 – Criação de um formulário para a tabela Livro

O formulário pode ser posteriormente editado, de forma a reposicionar os campos. No canto superior esquerdo pode-se modificar o tipo de vista: formulário; esquema; estrutura. Na vista de formulário o utilizador interage com o formulário normalmente. Na vista de esquema e estrutura, pode-se editar o formulário, existindo mais opções na vista de desenho, sendo no entanto a vista de esquema mais simples.

A zona de baixo do formulário existe uma barra que permite a navegação pelos registos na tabela. Esta barra é comum a todos os formulários.

Como foi definida uma relação dos livros com as cópias de livros, é criada também neste formulário um sub-formulário com uma lista para editar as cópias dos livros que estão ligadas a este livro, como se pode ver na Figura 4-2.

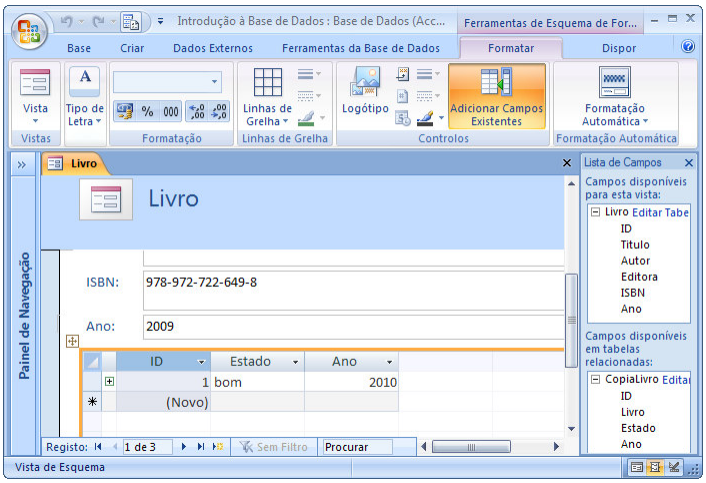


Figura 4-2 – Sub-formulário com as cópias existentes de um livro

Desta forma não só se pode neste formulário editar livros, como facilmente se pode adicionar cópias para este livro. Notar que não está presente o campo "Livro" da tabela CopiaLivro, porque esse é o campo utilizado de ligação, sendo mostrado no sub-formulário apenas as linhas que dizem respeito a este livro.

O formulário Livro foi criado automaticamente e está pronto. Não requer qualquer edição. É uma vantagem manter este (e qualquer outro) formulário intacto, já que se existirem alterações à tabela do Livro, facilmente se reconstrói o formulário. Se existir muita edição, por exemplo, reposicionamentos de campos, trocas de nomes e cores, quando for necessário alterar a tabela do Livro, tem que se actualizar este formulário.

O formulário Leitor, após criado fica com o aspecto da Figura 4-3.

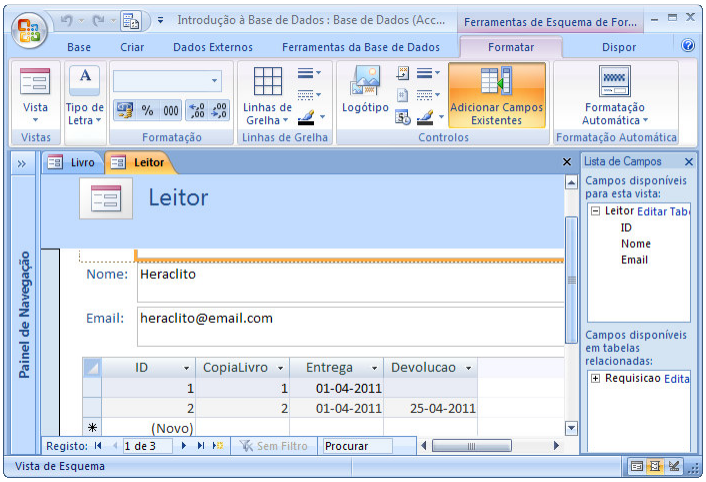


Figura 4-3 - Formulário automático para a tabela Leitor

No sub-formulário está apenas o código da cópia do livro. Naturalmente que seria mais interessante ter o título do livro, e não apenas o código da cópia do livro. Pretende-se editar o formulário para colocar a lista de livros lidos juntamente com as respectivas cópias. Para tal tem que se editar

primeiro a consulta de livros lidos, de forma ter não só as colunas que estão no sub-formulário, como também o nome do livro.

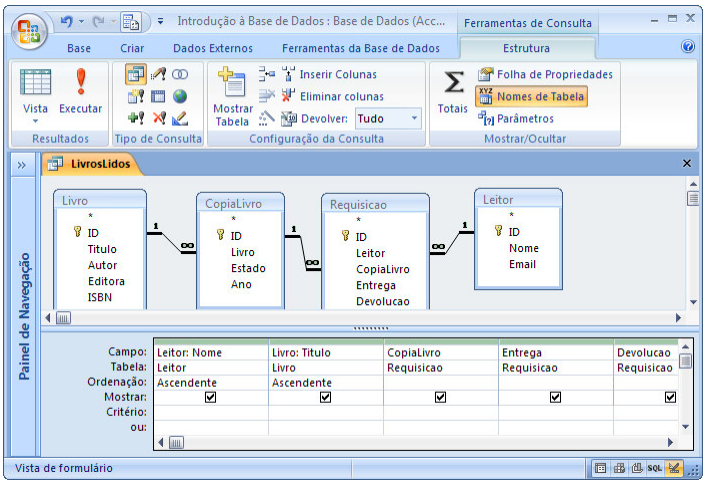


Figura 4-4 – Consulta dos livros lidos

Para fazer alterações ao formulário Leitor, este tem de ser colocado na vista estrutura, e editar o sub-formulário (seleccionar objecto » menu de contexto » propriedades), para substituir a tabela Requisicao pela consulta de LivrosLidos. A coluna de ligação com o formulário principal passou a ser o "Nome" que tem de coincidir com a coluna no formulário "Leitor". Pode-se ver a edição na Figura 4-5.

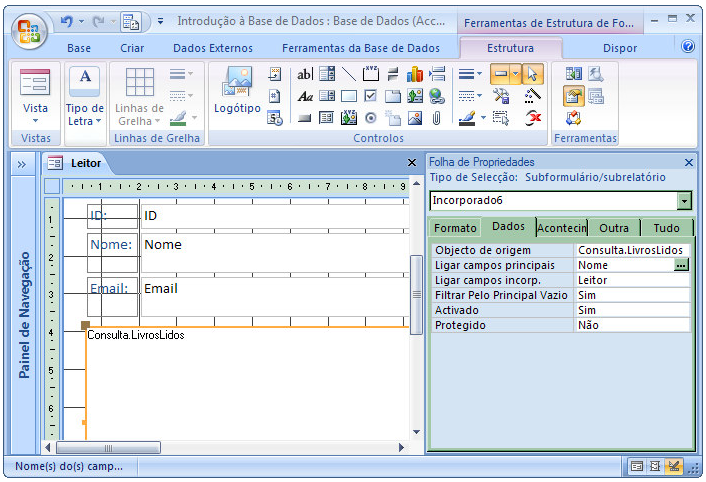


Figura 4-5 – Substituição da consulta associada ao sub-formulário

O resultado pode ser visto na Figura 4-6.

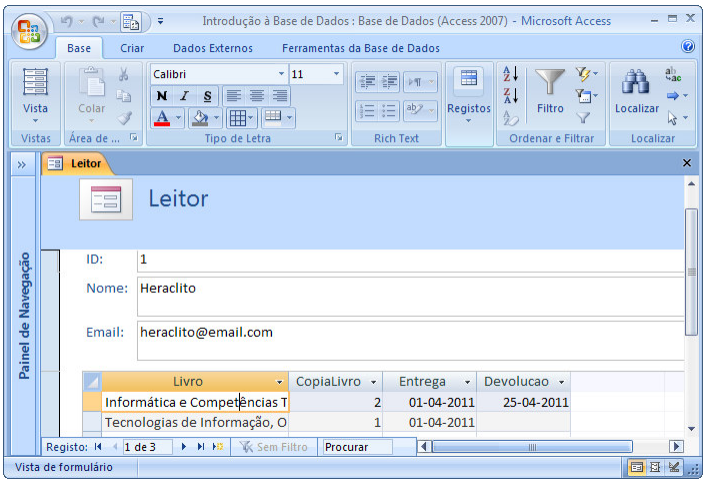


Figura 4-6 – Resultado da alteração do sub-formulário pela consulta

O nome do livro passa a ser visualizado. Alternativamente poderia ter sido apagado o sub-formulário e colocado num novo tendo como base a consulta desejada. É também possível ter mais que um sub-formulário dentro de um formulário.

É boa prática a colocação de sub-formulários com informações úteis para quem esteja a editar um determinado formulário. Neste caso, o utilizador ao editar o registo de um leitor sabe desde logo os livros lidos. No entanto, se houver demasiada informação para editar, é natural que o utilizador não dê tanta atenção a cada parte do formulário, pelo que é necessário optar por colocar apenas a informação relevante.

Embora não seja aconselhado demasiados formulários de forma a não distrair o utilizador, no nosso exemplo vamos criar formulários para as restantes tabelas de forma a exemplificar outras funcionalidades importantes nos formulários do *MS Access*. Um formulário de CopiaLivro criado a partir da tabela fica neste estado da Figura 4-7.

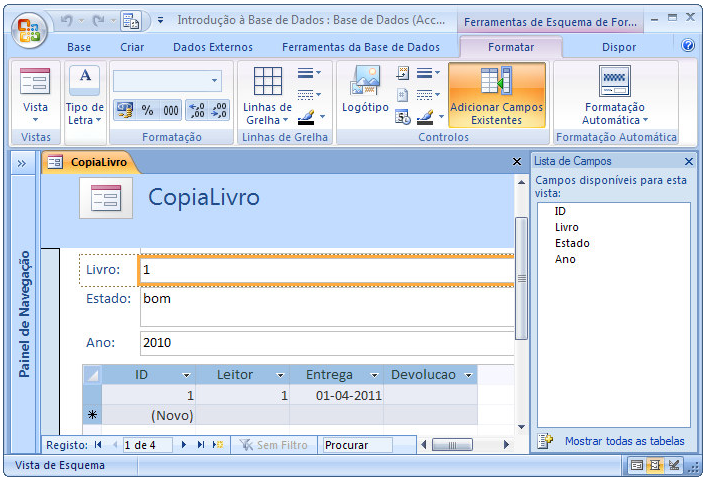


Figura 4-7 – Formulário da tabela CopiaLivro

Aqui temos dois problemas. O primeiro é o código do leitor em vez do seu nome, que seria certamente mais útil. O segundo é o código do livro em vez do seu título. Relativamente ao primeiro

problema já foi resolvido no formulário anterior, basta que se utilize a consulta de livros lidos. A consulta deve ser reutilizada, mas a configuração muda conforme o formulário em que a consulta está, como se pode ver na Figura 4-8.

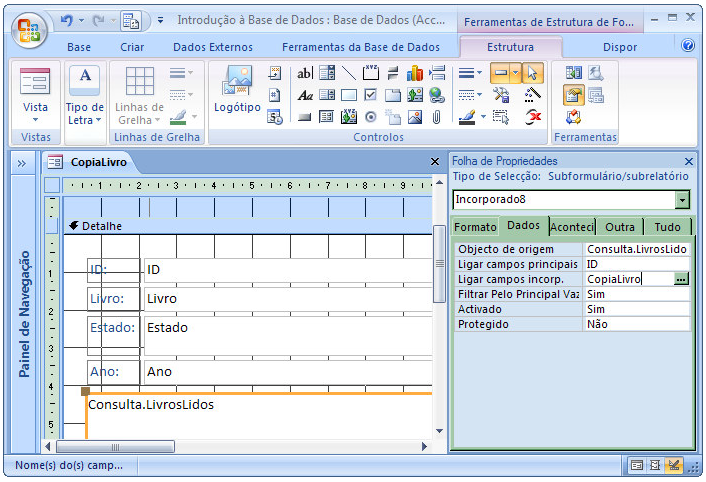


Figura 4-8 – Colocação da consulta LivrosLidos no sub-formulário CopiaLivro

A coluna com o título do livro pode ser ocultada no sub-formulário dado que essa informação já pertence ao formulário principal, e neste caso o campo a coincidir é o ID da tabela CopiaLivro com o campo da consulta CopiaLivro.

Resta-nos o segundo problema, editar no formulário principal não o código do livro mas uma lista com os títulos dos livros. Na vista de estilo pode-se adicionar mais campos "Adicionar Campos Existentes" o que permite abrir a lista de campos não apenas para a tabela associada ao formulário, como também para as tabelas relacionadas, como é o caso da tabela Livro, como se pode ver na Figura 4-9.

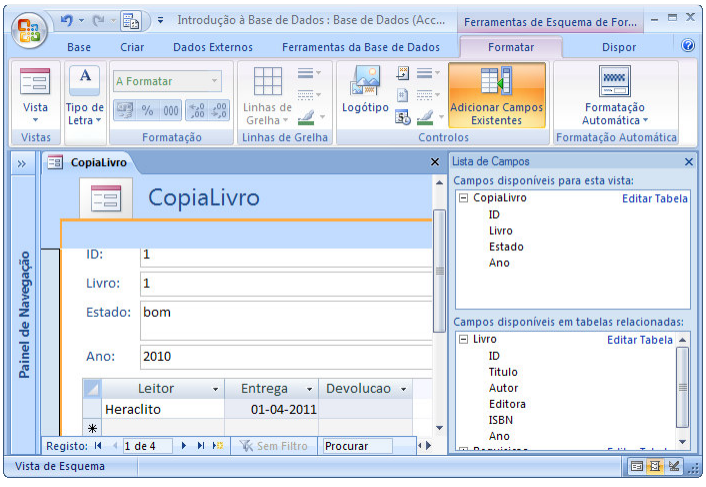


Figura 4-9 – Adicionar mais campos de tabelas relacionadas

Ao colocar a coluna "Titulo" no formulário, a tabela Livro passa a pertencer também ao formulário e pode-se apagar o campo "Livro" com um número e substituir pelo título do livro.

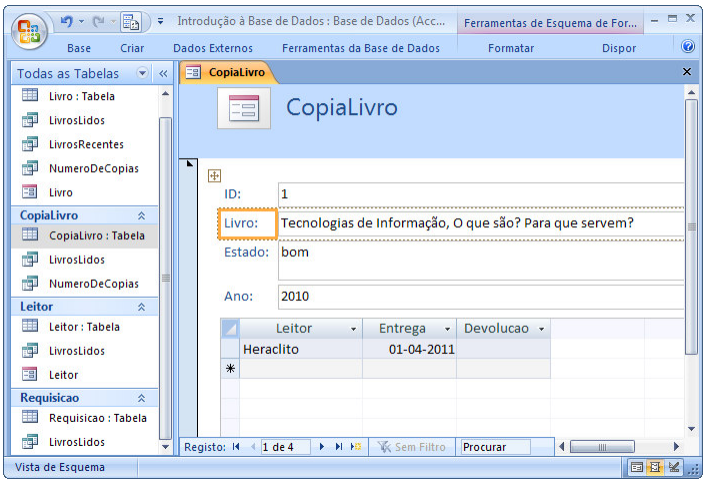


Figura 4-10 – Troca do campo numérico Livro, pelo título do livro

Falta um último formulário sobre a tabela Requisicao. Neste formulário vamos utilizar o formulário dividido: Criar/Formulário Dividido. O resultado pode ser visto na Figura 4-11, e fala por si.


Figura 4-11 – Criação de um formulário dividido

Este tipo de formulários tem não só uma vista de formulário para edição de um registo, como tem também a lista de todos os registos como se estivesse a editar a tabela, de forma a facilitar a navegação na tabela. Em todo o caso, a barra de navegação na zona inferior mantém-se, mas é incomparavelmente menos rápida que a vista tabular.

Este formulário tem o problema dos códigos no campo do leitor, e da referência à cópia do livro sem indicação do título do livro.

Para colocar o nome do leitor por troca com o código, utiliza-se a mesma técnica que se utilizou no código do livro no formulário de CopiaLivro, como se pode ver na Figura 4-12.

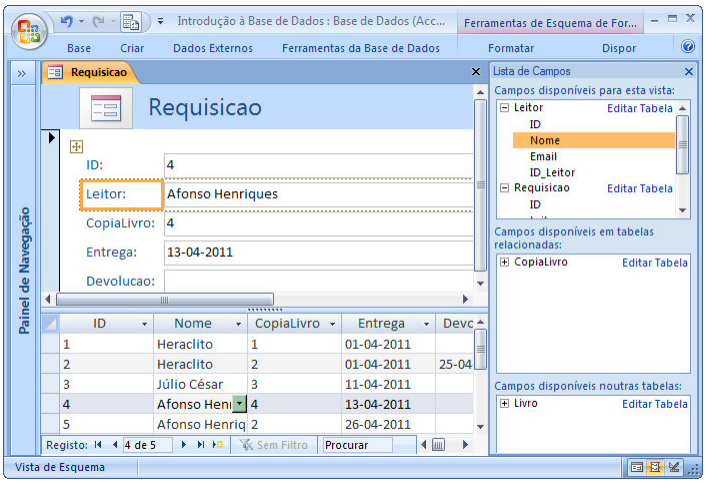


Figura 4-12 – Colocação do nome do leitor em vez do código, no formulário Requisicao

Para colocar o título do livro, tem que se colocar primeiramente a coluna "Livro" da tabela CopiaLivro, e de seguida a coluna "Titulo" da tabela Livro, substituindo a coluna "Livro". Desta forma ficam todas as tabelas neste formulário, cuja versão final, antes de apagar a coluna “Livro”, pode ser vista na Figura 4-13.

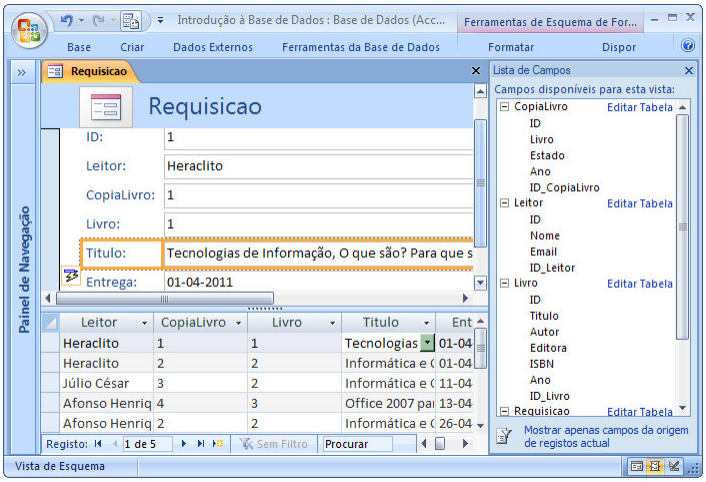


Figura 4-13 – Versão final do formulário Requisicao, com informação de todas as tabelas

Neste exemplo os formulários podem ainda ser melhorados, para além de ocultar os campos não necessários, pode-se por exemplo validar a informação. Não deixar emprestar uma cópia de um livro que não esteja devolvido, por exemplo. Pode-se adicionar condições de validação em qualquer campo, mas condições mais complexas acabam por requerer programação, pelo que ficamos por aqui no que toca a formulários.

# 5. RELATÓRIOS

Para fazer relatórios é suficiente seleccionar a consulta respectiva e se executar Criar/Relatório. Com base na consulta o relatório é criado. Por exemplo a consulta dos livros lidos fica com o aspecto da Figura 5-1.

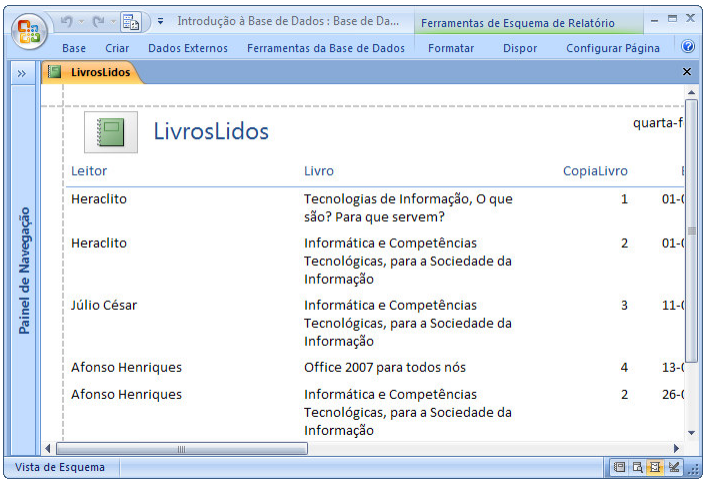


Figura 5-1 – Relatório associado à consulta LivrosLidos

Na vista de estrutura pode-se editar não só os campos dispostos, como cabeçalhos e rodapés existentes no relatório, como se pode ver na Figura 5-2.

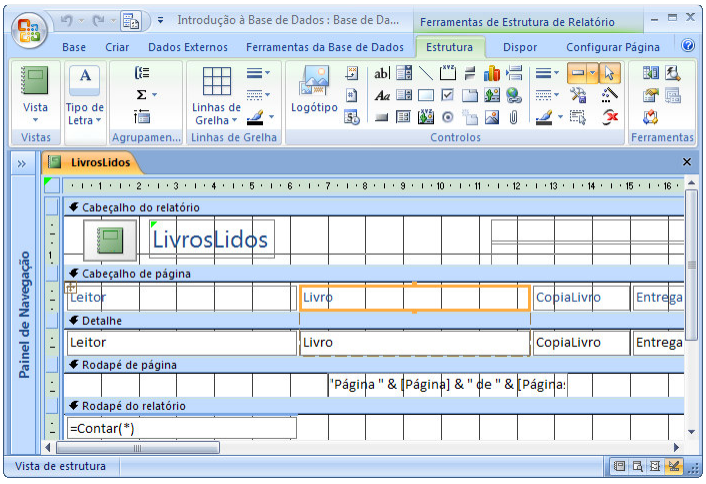


Figura 5-2 – Vista de estrutura do relatório da Figura 5-1

Este relatório está pronto a imprimir. Embora tanto numa consulta como num formulário possa estar informação relevante, nos relatórios essa mesma informação é colocada de forma a poder facilmente passar ao papel, e não pode ser editada.

Convém ter relatórios que identifiquem possíveis situações que gerem acções. Por exemplo, a lista de cópias entregue mas não devolvidas. Tem que se primeiro efectuar uma consulta que liste essas cópias, e de seguida fazer um relatório sobre essa consulta.

A situação pretendida é uma restrição à consulta que já existe dos LivrosLidos. Tem que se restringir esses resultados aos que não têm data de Devolucao atribuída. No entanto não se deve alterar a consulta LivrosLidos. Esta consulta está a ser utilizada em dois sub-formulários e em um relatório. Deve ser criada nova consulta.

Este é um erro comum, alterar algo e dias depois descobre-se que outra parte do sistema não está a funcionar como devia. No entanto fazer tudo de novo é também um erro, desaproveita o trabalho e atenção tomado para construir a consulta LivrosLidos.

Em vez de alterar ou fazer de raiz, pode-se fazer uma consulta baseando a nova consulta, LivrosPorDevolver, na consulta que tem praticamente tudo o que se pretende, restringindo apenas as suas linhas. Isto é possível porque nas consultas podem-se adicionar não só tabelas que façam parte da consulta, como outras consultas, potenciando assim a reutilização das consultas.

Se não existir data de devolução, o livro é considerado não devolvido. Esta consulta pode ser vista na Figura 5-3.

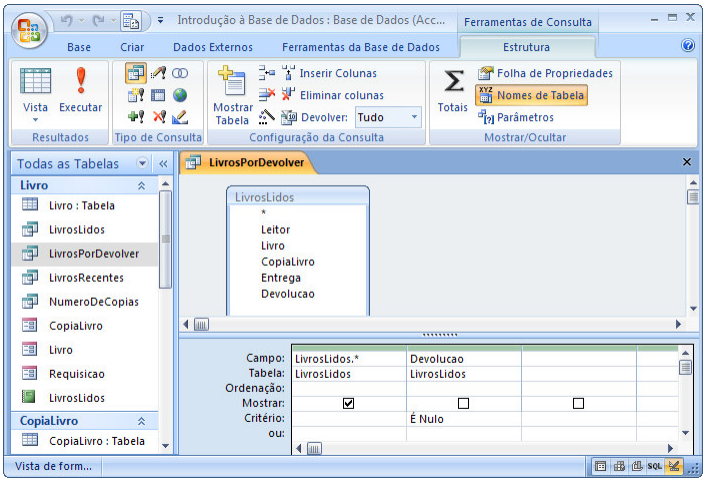


Figura 5-3 – Consulta LivrosPorDevolver baseada na consulta LivrosLidos

Pode-se agora criar um relatório com base nesta nova consulta, com toda a informação necessária. O resultado é apresentado na Figura 5-4.

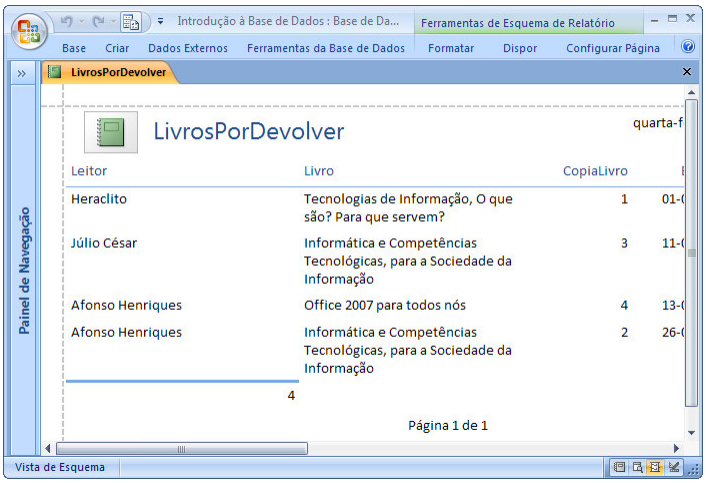


Figura 5-4 – Relatório resultante dependente da consulta na Figura 5-3

Outras consultas podem ser feitas, mas também neste capitulo, tal como nos formulários, não se pretende efectuar relatórios muito complexos. Esta é toda a matéria que necessita de saber para construir pequenas base de dados pessoais.

# 6. EXERCÍCIOS

É muito importante que pratique criando base de dados diversas, de modo a poder tomar as opções mais acertadas quando estiver a construir uma base de dados de interesse pessoal. É com o objectivo de dar uma base de prática que se segue um conjunto de exercícios para implementar no MS Access.

Cada exemplo possui duas versões, a primeira é a informação a colocar numa primeira fase de utilização da base de dados, e a segunda versão é a informação que se pretende passar a colocar numa segunda fase. Este tipo de situação é muito comum, uma base de dados se entra em utilização facilmente se vê informação que poderia também estar a ser guardada, outros relatórios que seriam úteis, etc. Por outro lado, não vale a pena fazer uma base de dados muito completa e depois os utilizadores acharem que é muita informação ou muito complexa e não utilizarem. Mais vale começar devagar, e se realmente for útil, então ter uma evolução. Se a primeira versão estiver feita de forma correcta, as actualizações serão fáceis de se fazer, caso contrário será mais complicado, sendo esta a componente que se pretende simular. Não deve portanto ver o que é pedido para a segunda versão enquanto não tiver implementada a primeira versão.

Nos enunciados dos exercícios é apenas descrita a informação/objectivo da base de dados. As tabelas necessárias, consultas, formulários, e relatórios que são úteis para a base de dados, fazem parte do exercício. Deve ser colocada alguma informação fictícia para clarificar a utilização dos diversos elementos da base de dados.

Deixa-se alguns conselhos:

- Não se deixe iludir pela natureza da base de dados: "Nunca precisaria duma base de dados deste tipo, pelo que não preciso pensar em como implementar este exemplo". Nada mais de errado. Apenas a diversidade lhe permitirá ter experiência necessária para assegurar as melhores opções nas base de dados que lhe serão úteis.
- Não veja uma base de dados idêntica antes de iniciar a reflexão sobre o que necessita. Se olhar para a estrutura de uma base de dados será influenciado pelas opções tomadas e não terá liberdade para tomar as opções que considera melhor. Tem tempo de ver outras base de dados após ter uma estrutura da sua base de dados para a qual esteja satisfeito, e reflectir sobre o que pode ser melhorado na sua base de dados.

## **Exercício 1:** Contactos

Versão 1: Pretende-se uma BD pessoal com a lista de contactos para cada conhecido, incluindo nome, contactos (email, morada, telefone, etc.), tipo de relação, data de nascimento/aniversário, etc.

Versão 2: Pretende-se numa segunda fase manter um registo de empréstimos de diversas naturezas, bem como adicionar a hipótese de introduzir novos tipos de contactos.

## **Exercício 2:** Escola

Versão 1: Pretende-se uma BD que registe numa escola cursos, disciplinas, turmas, alunos e notas finais às disciplinas.

Versão 2: Numa segunda fase pretende-se registar também as salas e horários.

### **Exercício 3:** Banco

Versão 1: Pretende-se um sistema que registe clientes, contas, movimentos e dê o saldo das contas.

Versão 2: Pretende-se numa segunda fase permitir contas com mais que um cliente.

### **Exercício 4:** Empresa

Versão 1: Pretende-se uma BD para uma empresa em que se possa registar os empregados, departamentos, cargos e ordenados.

Versão 2: Numa segunda fase pretende-se registar as horas de entrada e saída de cada funcionário.

### **Exercício 5:** Aluguer de automóveis

Versão 1: Pretende-se uma BD para uma empresa de aluguer de automóveis. O sistema deve registar carros, gamas, clientes, alugueres, e custos.

Versão 2: Numa segunda fase pretende-se contabilizar os custos de aquisição e manutenção de cada automóvel, bem como o número de quilómetros. Cada modelo deve fazer revisões com os respectivos custos de X em X quilómetros, existindo também uma quilometragem máxima após a qual o automóvel é vendido a preço residual.

### **Exercício 6:** Futebol

Versão 1: Pretende-se um sistema que registe as equipas, jogos, pontuação, golos marcados e sofridos.

Versão 2: Numa segunda fase pretende-se também a contabilização dos jogadores que marcaram os golos, e dos cartões recebidos por cada jogador em cada jogo. Pretende-se ainda expandir o sistema de forma a possibilitar mais que um ano, e mais que uma divisão.

### **Exercício 7:** Formula 1

Versão 1: Pretende-se o registo de equipas, pilotos, corridas e pontos.

Versão 2: Numa segunda fase pretende-se expandir o sistema de forma a registar vários campeonatos, e a registar a melhor volta nas pistas.

### **Exercício 8:** Comboio

Versão 1: Pretende-se um sistema para registar para uma só linha as diversas estações e respectivos horários.

Versão 2: Numa segunda fase considere várias linhas, e vários tipos de comboios.

## **Exercício 9:** Editora

Versão 1: Pretende-se um sistema para registar livros, edições e tiragem, autores, e livros em stock.

Versão 2: Numa segunda fase pretende-se expandir o sistema para contabilizar os custos de impressão e vendas.

## **Exercício 10:** Estudante

Versão 1: Pretende-se um sistema para que um estudante possa guardar todas as notas de todas as actividades de avaliação que realizou numa unidade curricular, bem como respectivas ponderações.

Versão 2: O sistema deve guardar a informação para todas as unidades curriculares que o estudante frequentou, com respectivos anos e agrupadas em cursos.