Escola Superior de Tecnologia de Viseu

Departamento de Informática

Informática II

Gestão Comercial e da Produção

Introdução aos Sistemas de Gestão de Bases de Dados Microsoft ACCESS

(TÓPICOS ABORDADOS NAS AULAS DE INFORMÁTICA II)

Curso de Gestão Comercial e da Produção
Ano Lectivo 2002/2003

Por:
Filipe Caldeira

ÍNDICE

1 O que é uma Base de Dados?

É possível dizer de uma forma genérica que qualquer conjunto de dados é uma Base de Dados (BD): uma agenda com as moradas de pessoas conhecidas, uma lista de CDs, um livro, apontamentos tirados nas aulas, os dados guardados nos computadores das Finanças sobre os contribuintes e a *World Wide Web*. O objectivo de criarmos e mantermos uma BD é a de poder obter e utilizar os dados lá guardados: procurar a morada de uma determinada pessoa, saber o que foi dito nas aulas sobre um tema ou procurar a página WWW do Prémio Nobel da Economia deste ano.
Embora sendo possível usar a definição genérica dada acima, o termo base de dados é aplicado hoje em dia principalmente para fazer referência a bases de dados informáticas, isto é, conjuntos de dados estruturados, manipulados usando um Sistema de Gestão de Bases de Dados (SGBD) ou *Database Management System* (DBMS). Para permitir ao utilizador atingir os objectivos referidos acima, um SGBD disponibiliza linguagens de:

- **definição de dados**: para criação e alteração da estrutura da BD (DDL - *Data Definition Language*)
- **consulta de dados**: obter e processar os dados armazenados (DQL - *Data Query Language*)
- **manipulação de dados**: para acrescentar dados novos e modificar dados existentes (DML - *Data Manipulation Language*).

Hoje em dia, cada vez mais SGBD, como o Access, "escondem" essas linguagens por trás de interfaces do utilizador gráfica.
Outras características dos SGBDs são:

- **acesso simultâneo**: vários utilizadores podem aceder e alterar a mesma BD ao mesmo tempo sem criar inconsistências. Por exemplo, 2 utilizadores diferentes podem consultar simultaneamente os dados do mesmo cliente. No entanto o SGBD não permite que ambos os utilizadores alterem esses dados ao mesmo tempo.
- **vistas**: diferentes utilizadores poderão ter o seu acesso limitado a partes da BD. Por exemplo, embora todos os dados de uma organização estejam na mesma BD, aqueles que são importantes para a definição da estratégia só podem ser consultados pela administração.
- **construção de aplicações**: a tendência actual dos SGBDs é para combinarem a gestão do armazenamento/manipulação dos dados com a construção das aplicações que implementam os processos da organização. Tradicionalmente, os processos eram implementados independentemente, com recurso a linguagens de programação mais ou menos integradas com o SGBD.

Alguns exemplos de SGBD de grande porte são ORACLE, Informix, Adabas, SQL Server e DB2. Para PCs temos o MySQL, Dbase, FoxPro e Access. Os primeiros têm mais capacidade e são mais fiáveis do que os últimos. Estes são adequados para uso doméstico, em pequenas empresas ou como forma de aceder a partir de PCs a BDs instaladas em sistemas de grande porte, através de uma aplicação acessível ao utilizador não especialista em informática

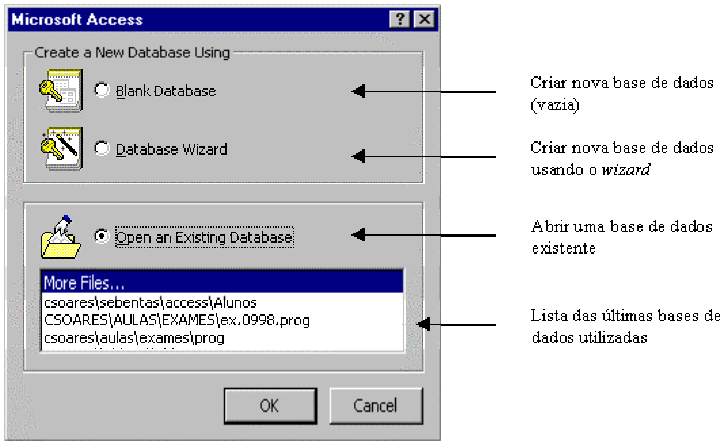
1.1 Porquê aprender Bases de Dados?

Porque é que alguém que não é (nem pretende ser) profissional de sistemas de informação deverá aprender a usar BDs?

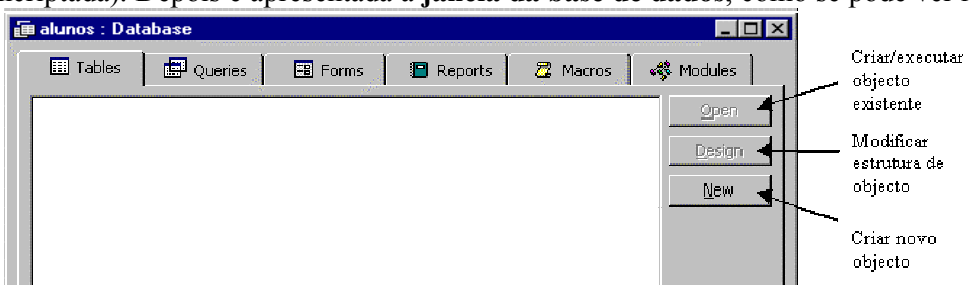
1. Pelo mesmo motivo pelo qual se devia aprender a usar um computador há 10 anos atrás: a divulgação crescente dos computadores fará com que dentro em breve os SGBDs sejam ferramentas de uso tão comum como são hoje em dia as folhas de cálculo.
2. Para facilitar a comunicação com os profissionais de sistemas de informação: a criação de sistemas de informação é um processo que envolve pessoas tanto da área de sistemas de informação como da área de acção da organização, sendo a comunicação entre ambos os grupos essenciais para o sucesso do sistema. Ora, os problemas de comunicação fazem com que o sistema criado raramente satisfaça as expectativas do cliente e mesmo algumas vezes seja o motivo do insucesso deste tipo de projectos. Sendo as BDs a infra-estrutura dos sistemas de informação, algum conhecimento destas poderá contribuir para melhorar a qualidade dos sistemas de informação.

2 O Sistema de Gestão de Bases de Dados Access

O Access é um Sistema de Gestão de Bases de Dados (SGBD) para PCs. Adequa-se ao uso doméstico, em pequenas empresas ou como forma de aceder a BDs instaladas em sistemas de grande porte.
Ao executar o Access é apresentada a seguinte caixa de diálogo:



Com o assistente (*wizard*) é possível construir uma base de dados (BD) a partir de um conjunto de modelos que podem ser alterados para melhor se adequarem ao problema em questão. Os modelos incluem BDs para listas de endereços, colecções de discos e livros, despesas, entre outras.
Se escolhermos criar uma BD vazia, é-nos pedido o nome do ficheiro onde ela será guardada. Cada base de dados do Access é guardada num único ficheiro, de extensão **mdb** ou (**mde**, no caso de ser uma BD encriptada). Depois é apresentada a **janela da base de dados**, como se pode ver na figura:



Atrás, definimos uma base de dados (BD) como sendo um conjunto de dados estruturados. No Access essa definição é estendida para englobar não só as estruturas, ou objectos, onde os dados são guardados mas também os objectos que permitem manipular esses dados.
Em cada separador da janela de bases de dados é guardado um tipo diferente de objectos:

- **Tables [Tabelas]**: objectos onde os dados são armazenados.
- **Queries [Consultas]**: objectos de consulta e processamento dos dados armazenados nas tabelas.
- **Forms [Formulários]**: estes objectos podem ser usados simplesmente como forma de dar um aspecto mais agradável e organizado às tabelas e consultas mas também como forma de ligação entre todos os objectos da BD, permitindo que o utilizador tenha a sensação de estar a trabalhar com uma aplicação e não com um conjunto de objectos.
- **Reports [Relatórios]**: objectos para formatar dados de forma a poderem ser imprimidos.
- **Macros**: objectos para automatizar acções.
- **Modules [Módulos]**: programas.

Cada objecto vai ser identificado por um nome dado pelo utilizador. Os objectos criados são guardados no ficheiro da BD.

2.1 A Ajuda

O Access, como a maior parte das aplicações Windows possui uma ajuda *on-line* muito completa. Está organizada como um livro mas também inclui um índice e um mecanismo de procura que facilitam a obtenção de informação sobre assuntos específicos.
O assistente do Office (*Office Assistant*) fornece ajuda relativa ao contexto. Por exemplo, se o utilizador estiver a fazer alguma coisa errada, ele procura "adivinhar" o que é pretendido e faz sugestões.

3 Distribuindo a Informação por várias Tabelas

Em aplicações mais complexas, a alocação da informação numa só tabela acarreta problemas de espaço e consistência.
Exemplo: Se um banco guardasse toda a informação sobre uma conta (número, data de abertura, nome do 1º titular, morada do 1º titular, etc) junto com a informação de cada movimento feito, o tamanho de cada registo seria enorme e a tabela atingiria uma dimensão incomportável. Para além disso, se, por exemplo, o 1º titular mudasse de residência, seria necessário corrigir todos os registos de movimentos na conta respectiva. Se se considerar o facto de que cada conta pode ter vários titulares, então o problema agravar-se-ia, dado que seria preciso acrescentar à estrutura da tabela campos para o nome e morada, etc dos outros titulares. Isto provocaria, entre outros problemas, desperdício de espaço, dado que nas contas com menos titulares do que o máximo previsto teriam vários campos vazios, e também dificuldade na definição da estrutura da tabela, para decidir qual o número máximo de titulares que uma conta pode ter.
Exemplo: Pretende-se registar informação sobre alunos, disciplinas e notas.

Notas : Table

Número	Nome	Morada	Localidade	Data de Nascimento	Disciplina	Regente	Nota	Data Lançamento
1	Carolina Santos	R. Cedofeita, 14	Porto	11-dez-79	Informática	C. Moura	15	10-jul-97
2	Maria Oliveira	R. Igreja, 3	Lisboa	23-mar-78	Econometria	P. Bravo	17	12-set-98
5	Ana Oliveira	Pç. Município	Moura	02-fev-79	Informatica	C. Moura	12	10-jul-97
6	Carolina Santos	R. Cedofeita, 14	Porto	11-dez-79	Econometria	P. Bravo	11	12-set-98
8	Maria Oliveira	R. Igreja, 3	Lisboa	23-mar-78	Informática	C. Moura	13	16-fev-98
(Number)			Porto				10	

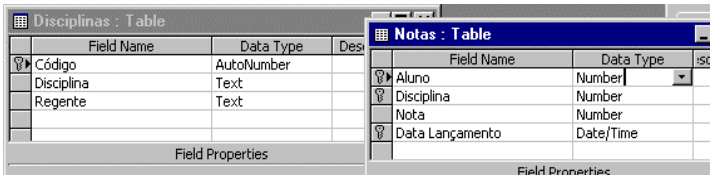
Pode-se verificar aqui redundância, com dados como a morada, localidade e data de nascimento a serem registados para cada nota do aluno. O mesmo sucede com o regente da disciplina que é guardado com cada nota da disciplina respectiva. Para além das dificuldades com a alteração de informação, a introdução repetida dos mesmos dados vai multiplicar os erros. Note-se, por exemplo, que na 4ª linha, o utilizador esqueceu-se do acento na palavra Informática. Este erro reflectir-se-ia, por exemplo, quando se calculasse a média das notas lançadas a 10 de Julho de 97 a essa disciplina. Uma solução seria preencher estes campos apenas uma vez, o que implicaria um desperdício de espaço considerável. Implicaria também que se esse registo fosse apagado seria necessário reintroduzir esses dados noutro registo.
Repare-se também que, como foi utilizado o tipo AutoNumber para o campo número, cada vez que é introduzida uma nota é atribuído um número diferente ao aluno.
Finalmente, como esta tabela tem informação de notas, seria impossível obter informação sobre um aluno que ainda não tivesse nenhuma nota lançada.

3.1 Relacionamentos

A melhor solução para estes problemas é dividir a informação por várias tabelas, relacionando-as para poder ter acesso à informação como um todo. O desenho da BD é a fase mais importante da criação de uma BD e resulta num **modelo de dados**.
Exemplo: A análise da informação a armazenar na BD da faculdade resulta num modelo de dados que se pode interpretar da seguinte forma:

- Há três entidades sobre as quais se guarda informação: alunos, disciplinas e notas.
- Cada aluno tem várias notas (à mesma ou a várias disciplinas).
- Para cada disciplina são lançadas várias notas (para o mesmo ou alunos diferentes).

Assim, para além da tabela Alunos original são criadas tabelas para as outras entidades representadas, Disciplinas e Notas:

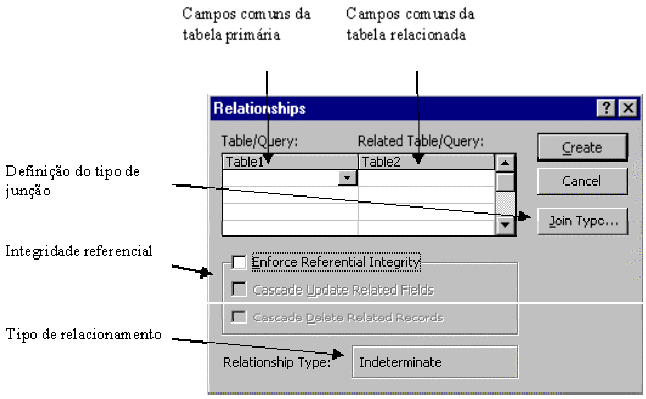


com os dados distribuídos pelas 3 tabelas da seguinte forma:

Note-se que a tabela Notas tem um campo Aluno que guarda o número do aluno a que a nota diz respeito. É acrescentado um campo chamado Código à tabela Disciplinas para substituir o campo Disciplina no papel de chave. A disciplina é, assim, identificada na tabela Notas pelo campo Disciplina que guarda um valor correspondente ao código.
Apesar da divisão da informação em várias tabelas, nada foi perdido: para saber o nome do aluno a que corresponde a nota na 3ª linha da tabela Notas, obtém-se o valor do campo Aluno desse registo, 2, e procura-se na tabela Alunos o aluno cujo Número é 2, que é a Maria Oliveira.
Se algum aluno mudar de morada, basta alterar o registo respectivo na tabela Alunos.
Os relacionamentos representados no modelo de dados são implementados directamente em Access. Um relacionamento envolve 2 tabelas que têm um **campo comum**, isto é, um campo que representa o mesmo elemento de informação. A criação de um relacionamento é uma indicação ao SGBD desse facto. O Access suporta dois tipos de relacionamentos:

- **um para um** ou **1:1** (**one to one**), em que a cada valor do campo comum corresponde um e um só registo em ambas as tabelas. Estes relacionamentos são utilizados, por exemplo, para separar informação sobre um determinado item à qual só uma parte dos utilizadores deve ter acesso.
- **um para muitos** ou **1:N** (**one to many**), em que a cada valor do campo comum numa das tabelas (**tabela primária** ou **lado 1** do relacionamento) correspondem vários registos, isto é, vários registos têm o mesmo valor nesse campo, na outra tabela (**tabela relacionada** ou **lado N**). Note-se que a cada registo do lado N corresponde 1 e um só registo do lado 1.

Os relacionamentos mais comuns são do tipo 1:N.
Para criar relacionamentos executar o comando Tools, Relationships. É apresentada uma caixa de diálogo para seleccionar as tabelas a relacionar:

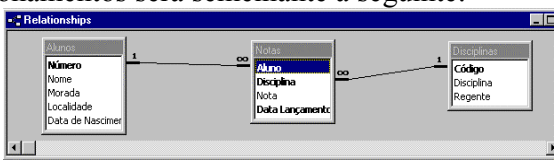


Seleccionadas as tabelas, o relacionamento é definido arrastando o campo comum na tabela primária para cima do campo comum na tabela relacionada. Um relacionamento pode envolver vários campos comuns, 2 a 2, bastando para isso preencher o resto das linhas disponíveis. O tipo de relacionamento é automaticamente detectado e indicado em baixo. O valor Indeterminate indica que o Access não conseguiu detectar o tipo de relacionamento.
O campo comum tem que ser do mesmo tipo em ambas as tabelas e ter também o mesmo tamanho (Field Size). A excepção à regra permite relacionar campos do tipo AutoNumber e Number, desde que o Field Size seja o mesmo.
O campo comum é, normalmente, o campo chave da tabela primária. Na tabela relacionada o campo comum é chamado **chave externa** (**foreign key**). No caso raro de haver um relacionamento entre tabelas em que o campo comum não é chave em nenhuma, então ele terá que ser indexado numa delas.
Exemplo: Para definir os relacionamentos da BD da faculdade, executar o comando Tools, Relationships e seleccionar todas as tabelas. Depois arrastar o campo Número da tabela Alunos para o campo Aluno da tabela Notas e o campo Código da tabela Disciplinas para Disciplina de Notas. O Access detecta que são relacionamentos 1:N.□
Para apagar um relacionamento é preciso seleccioná-lo e executar Edit, Delete. Atenção que a execução desse comando depois de seleccionada uma tabela só provoca o desaparecimento da tabela da janela de relacionamentos. Este mecanismo permite facilitar a análise dos relacionamentos em BD complexas. Para visualizar todos os relacionamentos executar Relationships, Show All e para ver todos os relacionamentos em que a tabela seleccionada participa, executar Relationships, Show Direct.
Numa BD bem desenhada e bem construída cada tabela está relacionada com pelo menos outra tabela.

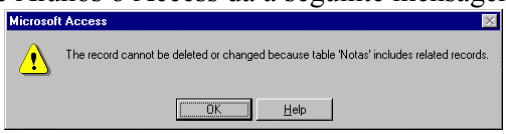
3.2 Integridade Referencial

Dividir a informação por várias tabelas resolveu os problemas de redundância e inconsistência referidos acima, mas criou outros. Por exemplo, se um registo numa tabela primária for apagado e existirem registos relacionados noutra tabela, a BD fica inconsistente. De forma semelhante, se for alterado o valor do campo comum de um registo da tabela primária, a informação na BD deixará de ser consistente.
Exemplo: Se eliminar o registo da tabela Disciplinas referente a Informática, a BD passa a estar inconsistente porque vários registos na tabela Notas fazem referência à disciplina 1 que já não existe. Isto é, não é possível, por exemplo, saber a qual disciplina é que a aluna Carolina Santos teve 15, como é indicado pelo 1º registo de Notas.
Se, por exemplo, os número dos alunos Maria Oliveira e Paulo Silva forem alterados na tabela Alunos (o que é impossível de fazer neste caso porque é um campo do tipo AutoNumber) então estes alunos passarão a ter as notas trocadas entre eles.
Activando a opção Enforce Referential Integrity, o Access passa a validar os valores dos campos relacionados, de forma a garantir a **integridade referencial**, e, portanto, a consistência da informação

distribuída pelas tabelas. Assim, todos os valores existentes no campo comum da tabela relacionada terão que existir no campo comum da tabela primária. Na janela de relacionamentos, um relacionamento com esta opção activada passa a ter indicação do lado 1 e do lado N, este último representado por um símbolo de infinito ( ).
Exemplo: Activando a opção Enforce Referential Integrity para os relacionamentos da BD da faculdade, a janela de relacionamentos será semelhante à seguinte:



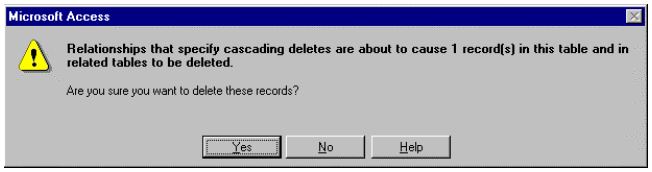
Se tentar apagar o 1º registo de Alunos o Access dá a seguinte mensagem de erro:



Note-se que a partir deste momento não é possível introduzir a informação na BD por qualquer ordem. Os dados das tabelas primárias têm que ser introduzidos antes dos dados das tabelas relacionadas.
Assim, para alterar o valor de um campo comum ou apagar um registo numa tabela primária, é necessário antes alterar/apagar os registos relacionados. Para simplificar estas tarefas que são frequentes, o Access permite que as alterações efectuadas na tabela primária sejam automaticamente reflectidas nas tabelas relacionadas após confirmação pelo utilizador.
Quando a opção Enforce Referential Integrity é activada são disponibilizadas mais 2 opções:

- **Cascade Update Related Fields**: provoca a propagação de alterações no campo comum do lado 1 a todas as ocorrências do valor alterado nos registos no lado N.
- **Cascade Delete Related Records**: provoca a eliminação de todos os registos do lado N com o mesmo valor do campo comum que o registo eliminado do lado 1.

Exemplo: Depois de activar ambas as opções de propagação em ambos os relacionamentos da BD da faculdade, se se apagar o registo da aluna Carolina Santos em Alunos, aparece a seguinte caixa de diálogo:



4 Elaboração de Modelo de Dados

4.1 Modelos de Dados

A informação que circula numa empresa, habitualmente sob a forma de documentos, é analisada por diferentes pessoas com diferentes perspectivas e objectivos. Por vezes tem que ser duplicada, outras vezes dividida de acordo com os interesses das pessoas que a vão analisar. Se pretendermos automatizar sectores da empresa, a informação tem que ser analisada de forma a obter um *modelo de dados,* que contemple os diferentes interesses dos utilizadores e que seja flexível de forma a possibilitar novas perspectivas.

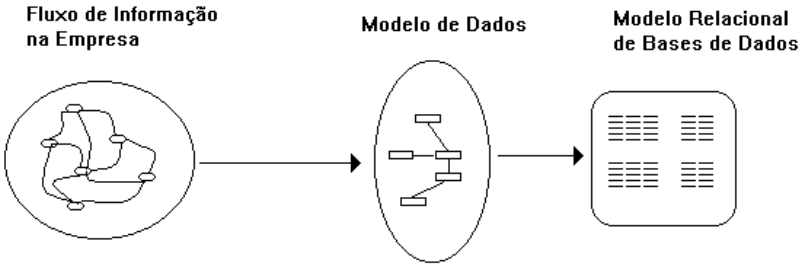
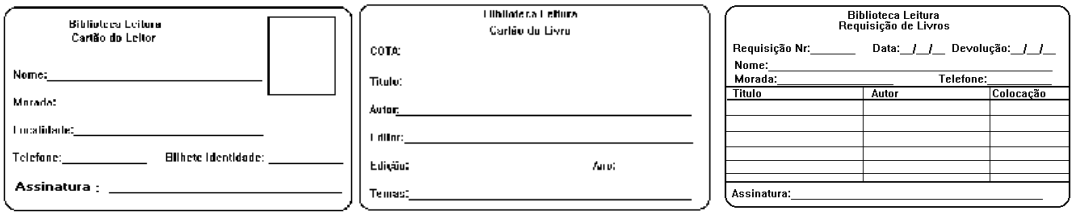


Fig. 1.

A implementação do modelo de dados, pode ser realizada com em *Sistemas de Gestão de Bases de Dados (SGBD)* relacionais. No modelo *relacional* de bases de dados, toda a informação é disposta em *relações (tabelas)*. Cada relação é caracterizada por um conjunto de propriedades ou *atributos*. Relações representam estruturas lógicas de dados que possibilitam uma boa comunicação entre utilizadores e o analista e facilmente implementáveis em computadores que suportem *Sistemas de Gestão de Bases de Dados*, de que o *Access* é um exemplo.

4.2 Uma Aplicação - Requisição de livros numa biblioteca

O nosso objectivo consiste em obter um conjunto de bases de dados que possibilitem guardar e manter a informação necessária para o bom funcionamento da biblioteca.
Supõe-se que, na interacção com o público, a biblioteca utiliza o documento da Fig. 2 (cartão de leitor) para identificar os leitores. Os leitores podem consultar os livros existentes na biblioteca utilizando os conjuntos de fichas (Fig. 3).
Existem três conjuntos de fichas (cada ficha está triplicada) ordenados de maneira diferente: um conjunto de fichas ordenado por ordem alfabética do *autor* do livro, o segundo ordenado por ordem alfabética do *título* do livro, e o terceiro por ordem alfabética do *tema* principal do livro.
Sempre que um leitor requisita livros, essa informação fica registada no livro de requisições, de que é mostrada uma página na Fig. 4.



Figs. 2, 3, 4.

Os documentos são uma fonte importante da informação e cuja análise é um bom inicio para a elaboração de um modelo de dados. Outra fonte de informação é o diálogo com os agentes por onde

circula a informação. A regra aqui é a de reunir o maior conjunto de informações sobre todo o ambiente envolvido. Qualquer modelo de dados corresponde a uma "interpretação" do analista em relação ao micro-mundo onde circula a informação. Uma interpretação incorrecta desse micro-mundo pode ter implicações inconvenientes, em termos de limitações do sistema.
Analisando o documento da Fig. 4, numa primeira análise, poderíamos ser levados a dispor a informação numa única base de dados, obtendo:

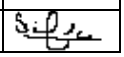
ReqNr	Data	Devol	Nome	Morada	Tel	Titulo	Autor	Coloca	Assina
1	7/6/94	14/8/94	Costa	R.de Cima	1234	"Dbase"	E.Jones	I.321	
						"Lotus"	D.Bolocan	I.124	
2	7/6/94	17/7/94	Silva	R.de Baixo	4771	"Dos"	M.Edward	I.567	
						"Dbase III"	J.Hall	I.322	

Fig. 5

Assim as duas primeiras linhas referem a requisição número *1*, feita em *7/6/94* pelo *Sr. Costa* que mora na *R.de Cima* com o telefone *1234*. Requisitou dois livros: um com o título *"Dbase"* escrito por *E.Jones*, cuja posição, na biblioteca é *I.321*; e o livro *"Lotus"* escrito por *D.Bolocan* que corresponde à posição *I.124*.
Esta disposição da informação, não está de acordo com os requisitos do modelo relacional de base de dados. Apesar de a informação estar disposta sob a forma de uma relação (tabela), existem atributos que não têm valor atribuído, pelo que a informação não é independente da ordem das linhas. Assim, se por qualquer processo (indexação, por exemplo) trocássemos a ordem das linhas, a informação ficava inconsistente. Uma forma de dispor a informação de acordo com o modelo relacional, será preencher completamente a tabela:

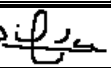
ReqNr	Data	Devol	Nome	Morada	Tel	Título	Autor	Coloca	Assina
1	7/6/94	14/8/94	Costa	R. de Cima	1234	"Dbase"	E.Jones	I.321	
1	7/6/94	14/8/94	Costa	R. de Cima	1234	"Lotus"	D.Bolocan	I.124	
2	7/6/94	17/7/94	Silva	R. de Baixo	4771	"Dos"	M.Edward	I.567	
2	7/6/94	17/7/94	Silva	R. de Baixo	4771	"Dbase III"	J.Hall	I.322	

Fig. 6

Uma análise desta BD evidencia a *redundância* da informação: sistematicamente a mesma informação aparece em diferentes linhas. Esta situação tem vários inconvenientes: por um lado, faz *crescer desmesuradamente* a BD o que implica maiores recursos em termos de capacidade de disco e velocidade de processamento. Por outro lado dificulta os processos de manutenção da BD. Imagine que um leitor muda de morada. Para manter a BD consistente, é necessário alterar a morada do leitor tantas vezes quantas as requisições por ele efectuadas, quando só deveria ser necessário alterar uma vez essa informação.
As situações de redundância da informação, manutenção complicada e também a falta de flexibilidade no sentido de obter outras maneiras de olhar para os dados que não aquelas previamente definidas, são as dificuldades sentidas quando os dados são analizados de uma forma primária. Estas situações são evitadas, obtendo um *modelo de dados*.
Vamos estudar duas metodologias, com o mesmo objectivo último: obter um conjunto de bases de dados que, por um lado, assegurem a consistência da informação, e por outro lado possibilitem guardar os factos o menor número de vezes possível.

4.3 Normalização

Este método representa um processo do específico para geral.

4.3.1 Formas Normais

Dois conceitos básicos, subjacentes a toda a análise baseada na normalização, são o conceito de *dependência funcional* e o conceito de *campo chave*.

No caso da Fig. 6, a cada número de requisição corresponde uma única data, mas o inverso já não é verdadeiro, isto é pode haver várias requisições efectuadas no mesmo dia. Dizemos que o atributo "data" é funcionalmente dependente do número de requisição. É usual representar como: *ReqNr Data*. Isto significa que *ReqNr* identifica a *Data*, ou seja, sendo conhecido o *número* da requisição podemos saber a *Data* da requisição.

De uma forma geral, o atributo A é funcionalmente dependente de B, se a cada valor de B corresponder um e um só valor do atributo A.

Chave é o atributo ou conjunto de atributos, cujos valores permitem identificar de forma única todas as linhas da relação. Eventualmente poderá haver situações em que haja várias possibilidades para a escolha de uma chave, e, num caso extremo, a conjunção de todos os atributos de uma relação constitui uma chave. O critério de escolha deverá ser baseado na *simplicidade*: é preferível uma chave com poucos atributos, a uma chave mais complexa. Numa análise da Fig. 6, a conjunção dos atributos ***ReqNr+Colocação*** constitui uma chave para a relação.

1ª Forma Normal

A informação da Fig. 6, satisfaz as condicionantes do modelo relacional: é dito estar *na 1ª Forma Normal (1ªFN).*

2ªForma Normal

Obtida a 1ª FN e identificada a chave da relação, vamos efectuar um estudo das dependências funcionais entre os atributos e a chave. A *data* da requisição, o *nome*, *morada* e *assinatura* da pessoa que requisita os livros dependem apenas do *número* da requisição e são independentes dos livros requisitados. Assim, teremos:

ReqNr, Data, Nome, Morada, Telefone, Assinatura

O título e o autor dos livros requisitados só dependem da *colocação* e são independentes do *número* da requisição. Teremos:

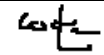
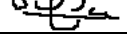
Colocação, Título, Autor

Em relação à *data* de devolução, o considerar que esta depende do número da requisição, implica que todos os livros terão que ser devolvidos simultaneamente, porque a cada requisição corresponde apenas uma data de devolução. Esta interpretação impõe limitações ao funcionamento da biblioteca. Seria mais funcional, se cada livro pudesse ser devolvido individualmente, ou seja:

ReqNr + Colocação, Devolução

A partir da relação original, na 1ª FN, obtemos três grupos de dependências dos atributos, em relação à chave. Isto significa que a relação da Fig. 6, pode ser decomposta em três relações:

Relação: **requisição**

ReqNr	Data	Nome	Morada	Tel	Assinatura
1	7/6/94	Costa	R.de Cima	1234	
2	7/6/94	Silva	R.de Baixo	4771	

Relação: **livros requisitados**

ReqNr	Colocação	Devolução
1	I.321	14/8/94
1	I.124	14/8/94
2	I.567	17/7/94

2	I.322	17/7/94

Relação : **livros**

Colocação	Titulo	Autor
I.321	"Dbase"	E.Jones
I.124	"Lotus"	D.Bolocan
I.567	"Dos"	M.Edward
I.322	"Dbase III"	J.Hall

Por decomposição da relação na 1ªFN, obtemos três relações, mais simples e com menos redundância. Por exemplo, a informação respeitante à pessoa que efectuou a requisição aparece apenas uma vez e é independente do número de livros requisitados. De notar que as relações mantêm atributos em comum: são estes atributos - chaves em relações- que vão permitir manter consistente a informação porque possibilitam estabelecer relacionamentos entre as relações.
ReqNr é a chave da relação ***requisição***, *ReqNr+Colocação* a chave da relação ***livros_requisitados*** e *Colocação* a chave da relação ***livros***.Nas três relações, todos os atributos dependem funcionalmente da chave: a informação é dita na 2ª Forma Normal.

3ª Forma Normal

Analisando a relação ***requisição***, todos os atributos dependem da chave, mas existem *dependências transitivas*, entre os atributos:

ReqNr, Nome, Morada

isto é, o atributo *Nome* depende funcionalmente da chave da relação, por sua vez o atributo *Morada* depende funcionalmente do *Nome*, o que implica que *Morada* dependa da chave da relação. Situação equivalente é a do atributo telefone. Desdobrando a relação, de forma a eliminar as dependências transitivas, obtemos duas relações ditas na 3ª FN.
Uma relação está na 3ª FN se cada atributo depender única e exclusivamente da chave:

ReqNr	Data	Nome	Assinatura
1	7/6/94	Costa	
2	7/6/94	Silva	

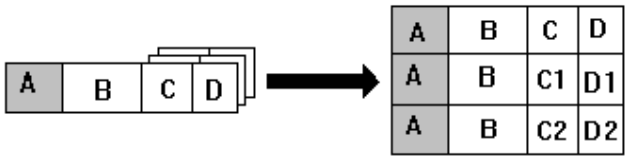
Nome	Morada	Tel
Costa	R.de Cima	1234
Silva	R.de Baixo	4771

4.3.2 Uma Solução para o Caso da Biblioteca

Normalizando os documentos relativos a leitores e livros (que deixamos como exercício para o leitor), é apresentada uma solução:
livros(colocação, título, autor, editor, edição, ano)
tema(tema,colocação)
leitores(bilhete_identidade, nome, localidade, telefone, fotografia, assinatura)
requisição(número, bilhete_identidade, data, assinatura)
reqlivro(número, colocação, data_devolução)

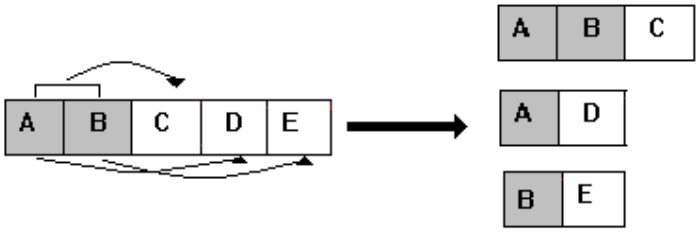
4.3.3 Esquema Geral da Normalização

Dispor a informação em tabelas de forma a que todas as ocorrências dos atributos sejam elementares (assumam apenas um valor).



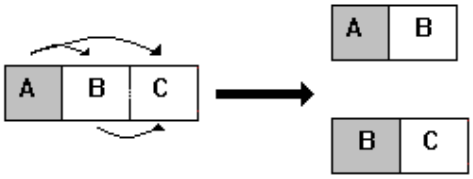
1ª Forma Normal

Identificar as dependências funcionais dos atributos em relação à chave. Desdobrar a relação de forma a que cada atributa dependa da totalidade da chave.



2ª Forma Normal

Identificar as dependências transitivas. Desdobrar a relação de forma a que cada atributo dependa única e exclusivamente da chave.



3 ª Forma Normal

4.3.4 ▾ Anexo A

Entidades e Relacionamentos

Este método representa um processo do geral para o específico.

Para problemas complexos, definidos por muitos atributos, os estados iniciais das relações, quando abordadas pelo método da normalização, são extensos e com dependências complexas. Torna-se fastidioso e sujeito a enganos, trabalhar com as relações não normalizadas.

4.3.5 Entidades

Um outro processo de obter um modelo de dados que satisfaz os objectivos inicialmente definidos, é a metodologia "Entidades e Relacionamentos". É uma metodologia que parte dos conceitos abstractos em jogo e os vai especializando. Por esse motivo, os passos intermédios são mais simples, envolvendo menos informação. O resultado final, é um conjunto de relações normalizado. (....)

▲

4.3.6 ▾ Anexo B

Tabelas

*Todos os dados que são armazenados de forma permanente pelo Access, são guardados em **tabelas**. Cada tabela armazena dados sobre um tipo de coisa, pessoa ou relação entre coisas/pessoas. Numa tabela pode-se, por exemplo, guardar informação sobre alunos, disciplinas ou notas (que podem ser vistas como relações entre alunos e disciplinas). As tabelas estão organizadas em linhas e colunas. Em cada linha guarda-se dados sobre um item, isto é, sobre um aluno, disciplina ou nota, respectivamente para os exemplos dados. Cada coluna representa uma determinada característica que está associada a todos os itens do tipo que a tabela representa. Por exemplo, numa tabela de alunos deverá existir uma coluna para o nome e outra para a data de nascimento. Em linguagem de bases de dados (BD), uma linha é um **registo** (record) e uma coluna é um **campo** (field). É possível ter um número potencialmente ilimitado de registos em cada tabela, mas o número de campos é fixo e limitado.*

*Cada campo de uma tabela tem um **tipo** associado que limita o tipo de dados que lá podem ser guardados. O campo "nome" da tabela de alunos será do tipo texto e o campo "data de nascimento" será do tipo data. Assim, no campo "nome"*

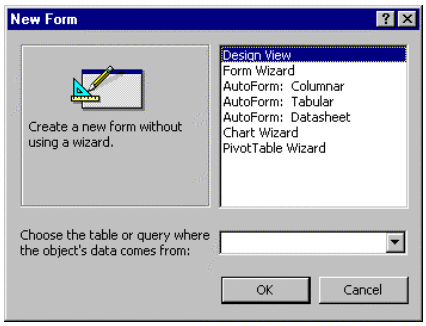
poderão ser introduzidos caracteres alfanuméricos enquanto no campo "data de nascimento" apenas podem ser guardadas datas. Ao conjunto de campos que compõe uma tabela, chama-se a ***estrutura*** *da tabela. O tipo determina também as operações que podem ser efectuadas com valores do campo. Por exemplo, não é possível fazer uma soma que envolva um campo do tipo texto.*
A informação inserida numa tabela é ***persistente****. Isto é, cada registo criado numa tabela é imediatamente guardado no ficheiro da BD. (...)*

5 Formulários

A visualização da informação nas folhas de dados (*datasheets*) das tabelas e consultas é pouco agradável e dificulta a sua manipulação. Os **formulários** servem para organizar e tornar mais agradável a interacção com a BD. Em conjunto com os controlos, permitem a construção de aplicações sobre a BD. Pode-se resumir as funcionalidades dos formulários a:

- editar (mostrar, alterar ou inserir) informação contida em tabelas (ou tabelas virtuais),
- mostrar informação derivada através de expressões (nos campos calculados),
- gerir a interacção com os utilizadores,
- mostrar informação suplementar (comentários, gráficos, etc).

Para criar um formulário, clicar no botão New no separador Forms da janela de BD. Aparece então uma caixa de diálogo que permite seleccionar o tipo de formulário a criar e a tabela ou consulta que servirá de fonte de registos:



As opções disponíveis são:

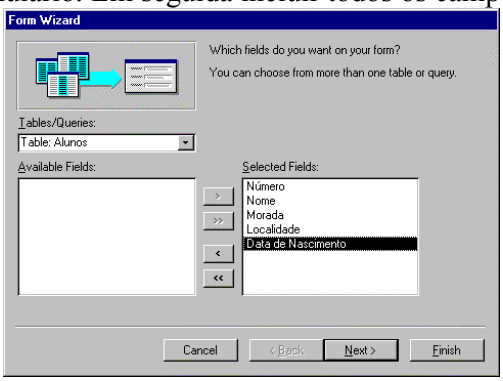
- **Design View [Vista de estrutura]**: Construção manual do formulário.
- **Form Wizard [Assistente de Formulários]**: Criação de formulários pela resposta a um conjunto de questões colocadas pelo assistente.
- **AutoForm**: **Columnar [Formulário Automático: Colunas]**: Esta opção e as 2 seguintes permitem a criação automática de formulários muito simples, baseados numa tabela apenas, em que a informação é organizada, respectivamente, um registo por página com os campos dispostos na vertical, …
- **AutoForm: Tabular [Formulário Automático: Tabelas]**: …vários registos por página com os campos dispostos na horizontal, …
- **Autoform: Datasheet [Formulário Automático: Folha de dados]**: …e em folha de dados.
- **Chart Wizard [Assistente de Gráficos]**: Criação de gráficos.
- **Pivot Table Wizard [Assistente de Tabelas Dinâmicas]**: Criação de formulários que apresentem os dados de forma agregada organizados em dimensões (*data cube*) através de Tabelas Dinâmicas (*Pivot Tables*) do Excel.

A maior parte das vezes, a melhor forma de construir um formulário é começar por usar o assistente para construir o essencial da visualização da informação. Depois, na vista de estrutura, efectuam-se pequenos ajustamentos e acrescentam-se controlos para associar funcionalidades ao formulário, como procura de registos, ligação a outros formulários, entre outras. A excepção mais comum a esta regra é a construção de formulários que servem como menus, em que se começa a construir um formulário do nada, usando a vista de estrutura.

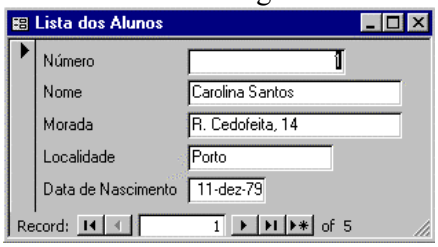
5.1 O Assistente

Para criar um formulário com o assistente, escolher a opção Form Wizard e, opcionalmente, seleccionar a tabela que lhe vai servir de base. Em seguida aparece uma caixa de diálogo para escolha dos campos a incluir. Note-se que nesta altura é possível alterar a tabela que serve de base ao formulário e é possível usar campos de várias tabelas. Neste último caso, o assistente guia o utilizador na criação de um formulário com sub-formulários, como será explicado mais adiante. Se for seleccionada apenas uma tabela é apresentada em seguida uma caixa de diálogo para escolher a organização dos campos, em que, para além das opções Columnar, Tabular e Datasheet explicadas acima, existe a opção Justified, em que os campos são dispostos uns a seguir aos outros, da esquerda para a direita e de cima para baixo. Na caixa de diálogo que se segue define-se o estilo a usar e na última, o nome do formulário e se se deseja abrir o formulário para ver os dados ou em vista de estrutura.

Exemplo: Para tornar a visualização e edição dos dados sobre os alunos mais agradável, cria-se um formulário. Clicar em New no separador Forms, seleccionar a tabela Alunos para fonte de registos e o assistente para criação do formulário. Em seguida incluir todos os campos de Alunos no formulário:

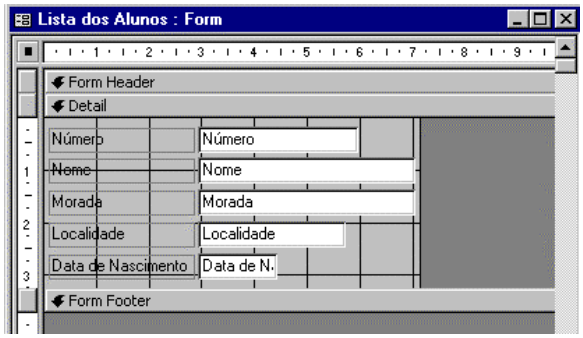


Escolher a organização dos campos em coluna e depois o estilo Standard. Finalmente, atribuir ao formulário o nome Lista de Alunos. O resultado é o seguinte:



5.2 A Vista de Estrutura

Como foi dito a utilização mais frequente da vista de estrutura é para editar formulários construídos com o assistente. Um formulário consiste em várias secções: **cabeçalho** (**Form Header**), **detalhe** (**Detail**) e **rodapé** (**Form Footer**).
Exemplo: Seleccionar o formulário Lista de Alunos e abrir a vista de estrutura:

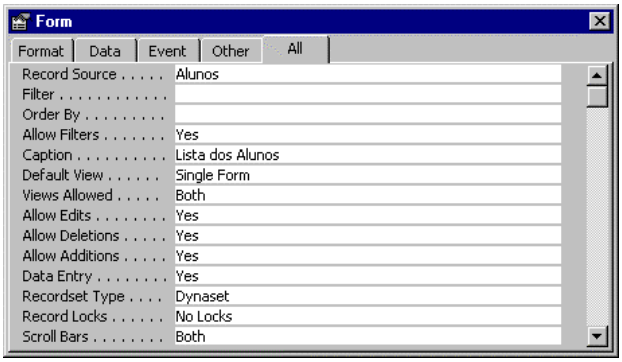


O cabeçalho serve para apresentar informação sobre o formulário, como o título. A zona de detalhe é normalmente usada para mostrar os campos do(s) registo(s). O rodapé é essencialmente usado para informação de estado, como, por exemplo, o número do registo actual, e informação agregada, como totais.
O utilizador pode ver as propriedades associadas a um formulário executando o comando View, Properties. As propriedades são agrupadas em 4 conjuntos:

- **Format**: formatação e aspecto visual.
- **Data**: qual a fonte de registos e como estão ligados.
- **Event**: associação de procedimentos/macros para interacção do utilizador com o formulário.
- **Other**: outras propriedades (menu, ajuda, impressão, etc).

O separador All apresenta todas as propriedades.

Exemplo: As propriedades do formulário Lista de Alunos são:



Note-se que o facto de o formulário basear-se na tabela Alunos, indicado ao assistente, foi registado na propriedade Record Source.

5.3 Controlos

As peças utilizadas para a construção de um formulário são os **controlos**. Para além de mostrar informação, os controlos permitem organizar e decorar o formulário, de forma a tornar a sua utilização mais agradável e eficiente. Um controlo pode ser colocado em qualquer uma das secções e tem também um conjunto de propriedades agrupadas da mesma forma que nos formulários. Alguns controlos têm assistentes que facilitam a sua criação.
A seguinte tabela apresenta os controlos existentes no Access:

Controlo	**Utilização**	**Assistente**	**Control source**

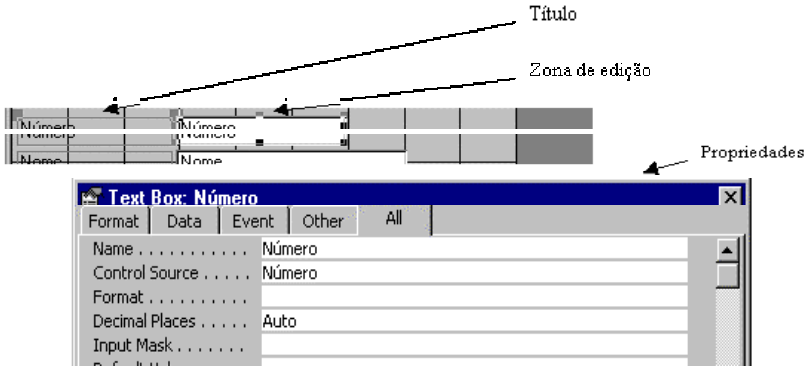
Label [Rótulo]	Informação ou texto descritivo.	Não	Não
Text Box [Caixa de texto]	Mostrar e/ou editar campos de tabelas e inserção de dados pelo utilizador, por exemplo, para procura de registos. É constituído por 2 partes, uma caixa com um título e outra onde se visualizam/editam os dados, e em que cada uma tem as suas propriedades.	Não	Sim
Option Group [Grupo de opções]	Pode conter *check boxes*, *option buttons* ou *toggle buttons*. Apenas uma opção pode estar seleccionada.	Sim	Sim
Toggle Button [Botão de alternar]	Se é utilizado isoladamente, selecciona um valor de Yes/No. Em *option group* selecciona um valor de um conjunto de valores.	Não	Sim
Option Button [Botão de opção]	Idem *toggle button*.	Não	Sim
Check Box [Caixa de verificação]	Idem *toggle button*.	Não	Sim
Combo Box [Caixa de combinação]	Selecção de valor de uma lista. Se não estiver aberta, apenas o valor escolhido é mostrado. Permite inserção de novos valores.	Sim	Sim
List Box [Caixa de listagem]	Selecção de um valor de uma lista. O controlo está sempre aberto. Não permite inserção de valores novos.	Sim	Sim
Command Button [Botão de comando]	Permite executar comandos, macros ou chamar procedimentos.	Sim	Não
Image [Imagem]	Imagem guardada em ficheiro.	Não	Não
Unbound Object Frame [Moldura de objecto independente]	Objecto OLE não ligado (unbounded).	Não	-
Bound Object Frame [Moldura de objecto dependente]	Objecto OLE ligado (bounded).	Não	-
Page Break [Quebra de página]	Separa um formulário em páginas.	Não	Não
Tab Control [Controlo separador]	Separador usado, por exemplo, para organizar de forma mais legível tabelas com muitos campos.	Não	Não
Subform/ Subreport [Subformulário/ subrelatório]	Inserir um subformulário ligado ao formulário actual, por exemplo, os movimentos de uma conta bancária cujos dados são apresentados no actual.	Sim	Não
Line [Linha]	Linha decorativa.	Não	Não
Rectangle [Rectângulo]	Rectângulo decorativo.	Não	Não
More Controls [Mais controlos]	Outros controlos disponibilizados pelo Access ou outras aplicações.	-	-

Das propriedades dos controlos, duas, comuns a vários, são de especial importância: Name e Control Source. Control Source permite definir a forma como o controlo obtém a informação que mostra:

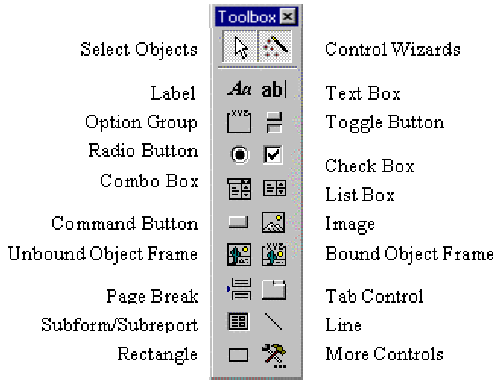
- Campo: Para ligar um controlo a um campo da fonte de registos introduz-se o nome deste nesta propriedade. A lista de campos (View, Field List) permite acrescentar controlos ligados a campos a um formulário por *drag&drop*.
- Expressão: É possível calcular expressões em formulários como se faz nas consultas. O conteúdo dos campos da fonte de registos são usados como nas consultas, isto é, o nome do campo entre parêntesis rectos ([*nome do campo*]). Para fazer referência a outros controlos, usar a seguinte notação [Forms]![*nome do formulário*]![*nome do campo*], em que o *Nome do campo* refere-se ao conteúdo da propriedade Name.
- Utilizador: Para obter dados ou parâmetros do utilizador deixa-se esta propriedade vazia ou com o valor Unbound.

Name indica o nome do controlo, permitindo utilizar ao seu conteúdo (isto é, o valor mostrado) numa expressão, macro ou num procedimento.

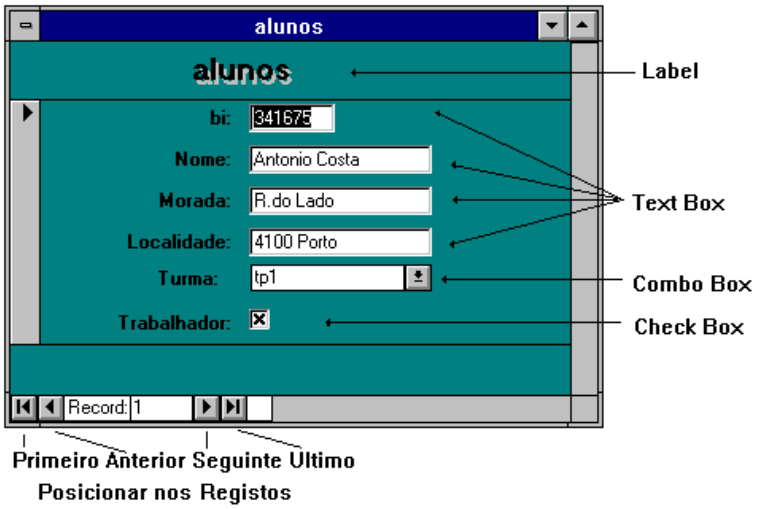
Exemplo: As propriedades de um dos controlos da Lista de Alunos são:



Para facilitar a edição de formulários, o Access disponibiliza uma barra de ferramentas (View, Toolbox):



O botão Select Objects [Seleccionar objectos] activa um apontador que permite seleccionar (por exemplo, para ver as propriedades), mover, alterar tamanho e editar objectos num formulário. O botão Control Wizards [Assistentes de controlos] liga/desliga os assistentes de controlos. A utilização de assistentes na construção de controlos é particularmente útil porque facilita as tarefas de lhes associar comportamentos, como procura de registos, abertura de formulários, etc. De outra forma, esses comportamentos só poderiam ser definidos usando macros e procedimentos em Visual Basic.
Para inserir um novo controlo clicar no botão respectivo e seleccionar, arrastando, a zona do formulário onde o controlo deverá ser colocado.
O seguinte formulário (tipo *columnar*) apresenta alguns destes controlos:

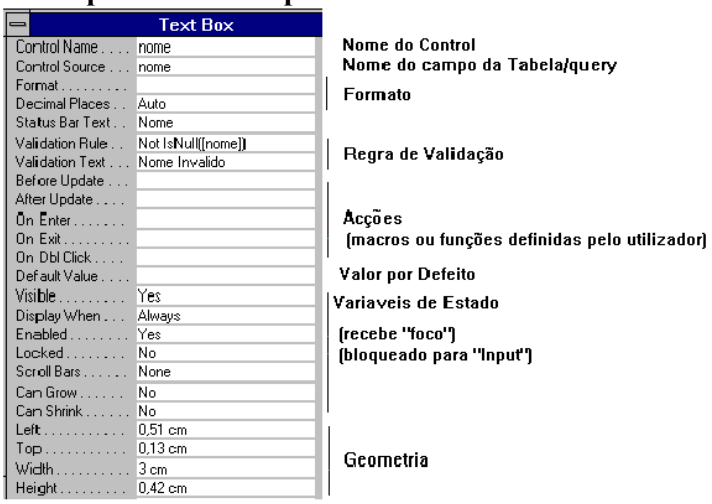


A seguir alguns dos controlos utilizado no formulário alunos serão analisados em pormenor.

5.4 Caixa de Texto para Nome

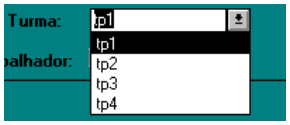
Consideremos a caixa de texto que contém o nome do aluno. A caixa de texto é identificada por **nome** (é o nome deste controlo) e está ligada ao campo nome da tabela alunos (ver propriedades deste controlo). Assim, se o utilizador tiver um registo em branco (não preenchido) e digitar um nome, esse nome será introduzido na tabela.
Neste caso a fonte de informação (*Control Source*) é associada a um campo da tabela alunos. Noutras circunstâncias a caixa de texto podia ser conectada à uma expressão, ou alternativamente ficar não ligada (para permitir ao utilizador digitar informação que não é armazenada).

Propriedades de controlo tipo "Text Box" para "nome":

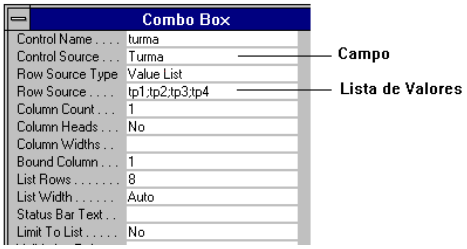


5.5 Combo Box para entrada da Turma

Suponha que o nome de aluno já foi preenchido, mas falta escrever os restantes dados. A introdução da turma pode ser feita através uma *Combo Box*, pois há um número limitado de alternativas. O seguinte quadro mostra o que se pretende obter:



Uma *Combo Box* pode ser construída com o auxílio de um *wizard* ou não. Aqui exemplificamos a sua criação manual. Para garantir que isso aconteça, deve ligar o controlo ao campo apropriado da tabela (campo Turma*)* especificando o *Control Source*. As opções (tp1; tp2 etc,) devem ser introduzidas no *Row Source* de propriedades:

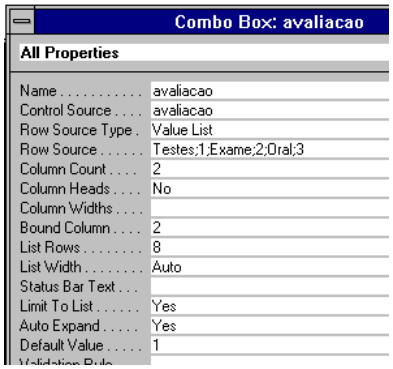


5.6 Introdução de um Grupo de Opções ("Option Group")

Como para a Combo Box, também os Option Group podem ser construídos com um *wizard*. Nesta secção consideramos várias alternativas para mostrar o número limitado e pré-definido de valores no formulário.

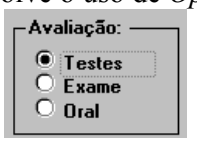
Suponha que pretende armazenar informação sobre o **tipo de avaliação** em que os alunos foram avaliados. Actualmente considere três possibilidades de avaliação, mutuamente exclusivas: **testes**, **exame final** ou **exame oral**.

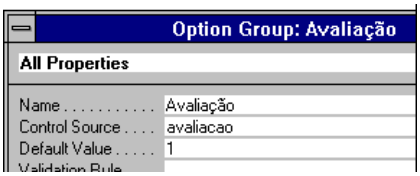
Primeiro consideramos algumas alternativas que não envolvem o grupo de opções. Uma possibilidade, é utilizar uma *Combo Box* com duas colunas:



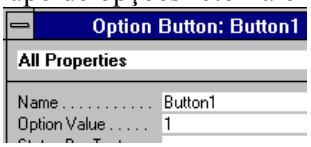
Uma outra possibilidade seria armazenar esta informação num campo tipo Text com uma regra de validação apropriada, por exemplo: IN("Testes";"Exame";"Oral").

No entanto, a solução com o ***grupo de opções*** é mais apropriada. Assim, deverá associar a cada opção um código, por exemplo 1 para "Testes", 2 para "Exame" e 3 para "Oral". Num grupo cada opção é apresentada por controlos que poderão ser Check Box, Option Button ou Toggle Button. O efeito é que assim será possível escolher só uma das diferentes alternativas. Considere, por exemplo, a seguinte solução que envolve o uso de *Option Buttons*.





Para criar um grupo de opções deste tipo, em primeiro lugar é criado o enquadramento, arrastando-o para o formulário da ferramenta.
Posteriormente são acrescentadas as opções, arrastando o controlo desejado (ex. *Option Button*), para dentro da caixa do enquadramento.
Ao acrescentar uma opção, à propriedade Option Value deste controlo é atribuído um número sequencial. A figura seguinte mostra as propriedade de Button1. O Option Value deste botão é 1. Quando o controlo é seleccionado, o grupo de opções retorna o valor de OptionValue deste controlo.

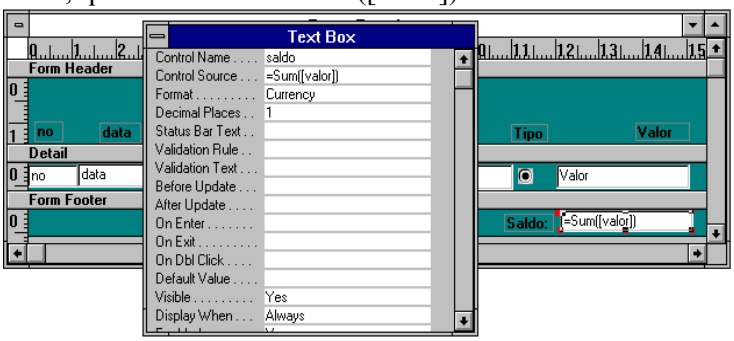


Ao escolher uma opção, ao campo da tabela ligada fica atribuído o valor de parâmetro Option Value do controlo escolhido. Assim ao campo avaliação fica atribuído o valor 1 se o utilizador escolher o botão testes; o valor 2, se escolher o botão exame, etc.

5.7 Cálculo de Totais

Suponha que se pretende calcular o total de todos os valores que aparecem no campo valor:

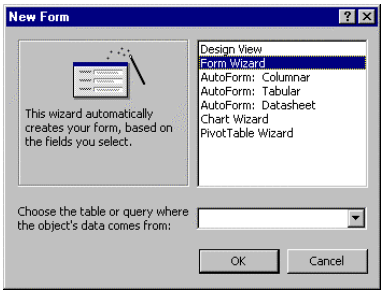
Para atingir este fim, deveria criar uma caixa de texto, cujo fonte de informação *(*Control Source) deveria ser uma expressão, que neste caso é =Sum([valor]):



Os campos que aparecem na expressão ligada ao controlo que efectua a operação sobre um conjunto de registos, deverão ser campos da fonte de registos (Tabela ou *Query*) (isto é, não controlos calculados no formulário).

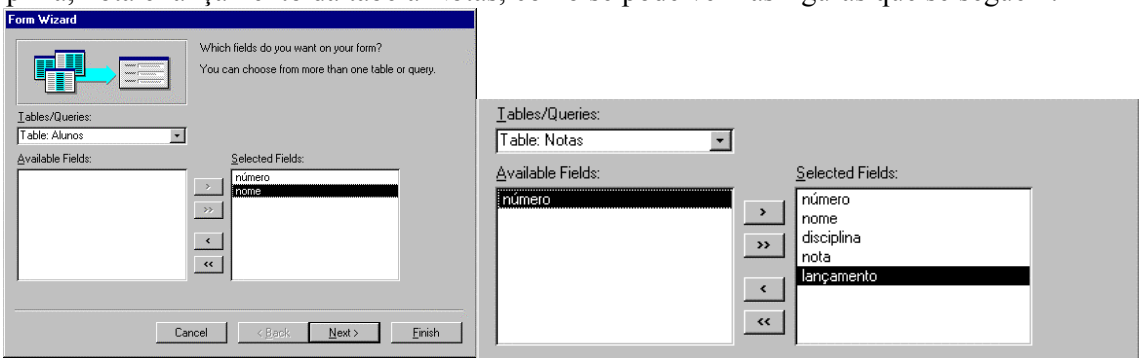
5.8 Criação de Main/Subform

Como já foi dito atrás é frequentemente necessário obter informação que não se encontra em apenas uma tabela. Para isso podemos criar uma *query* que envolva as várias tabelas pelas quais os campos desejados se encontram distribuídos e usar um formulário para visualizar os resultados da execução dessa *query*. Alternativamente podemos usar o mecanismo de formulário principal/subformulário (*main/subform*).



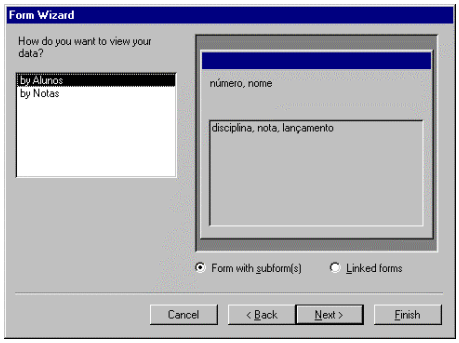
Por exemplo, suponhamos que queremos ver as várias notas de cada aluno usando um formulário do tipo formulário principal/subformulário. Para criar um formulário deste tipo usamos o *FormWizard*: cria-se um formulário (comando New no separador Forms) e selecciona-se a opção FormWizard, como se pode ver na figura. Note-se que não é necessário indicar a tabela/*query* de onde os dados são obtidos.

Em seguida acrescentar os campos desejados das várias tabelas/*queries* aos campos seleccionados (Selected Fields). Neste caso, acrescentamos os campos número e nome da tabela Alunos e os campos disciplina, nota e lançamento da tabela Notas, como se pode ver nas figuras que se seguem.



É importante notar que tem que existir um relacionamento **1:N** entre as tabelas/*queries* envolvidas para que o *wizard* funcione correctamente. Se tal não fôr verdade, o *Access* não será capaz de detectar que se pretende criar um formulário do tipo formulário principal/subformulário. Para dar a volta a este problema pode-se usar o controlo *subform* (ver Controlo subform).

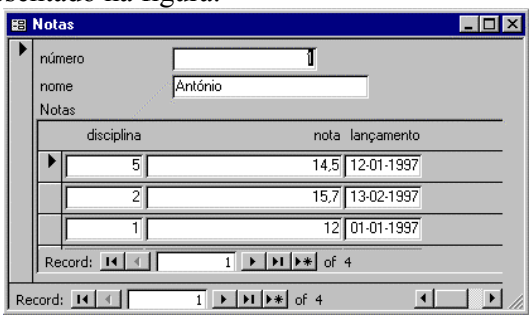
Em seguida dado que vamos visualizar informação de mais do que uma tabela, é necessário indicar a forma como queremos que essa informação seja organizada. Neste caso, como temos informação sobre alunos e notas, a questão será: queremos ver as notas de cada aluno ou os alunos de cada nota? A resposta é, obviamente, a primeira. Assim, a informação deve ser organizada por aluno (by Alunos), como se pode ver na figura.



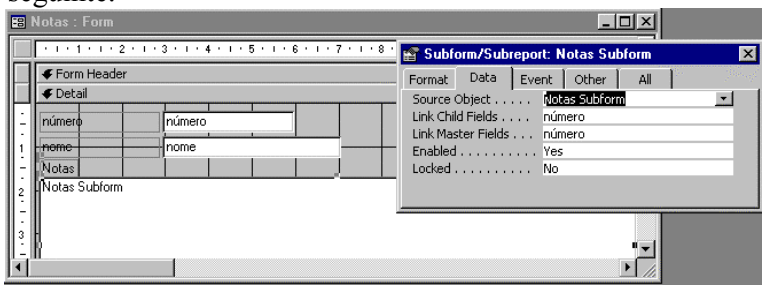
Em caso de dúvida deve-se experimentar as várias alternativas, podendo-se antever no lado direito da janela o aspecto do formulário em criação.

Na mesma janela é ainda possível escolher entre ver toda a informação numa só janela (Form with subform(s)) ou em várias (Linked forms). Escolhendo esta última, é criado um botão no formulário principal que dá acesso ao subformulário. Naturalmente, neste subformulário serão apresentados registos da tabela/*query* respectiva que estejam relacionados com o registo mostrado no principal. Isto é, no exemplo dado, se estivermos a visualizar o registo do aluno de nome António, clicando no botão serão apresentadas as suas notas. Mais uma vez, é possível antever o resultado da opção seleccionada.

Nas duas caixas de diálogo seguintes são apresentadas opções relativas ao aspecto gráfico do formulário. Finalmente, na última caixa de diálogo podemos dar nome aos formulários criados, resultando no formulário apresentado na figura.



Note-se que são criados 2 formulários, um com os dados da tabela Alunos e outro com os dados da tabela Notas. No formulário principal o *wizard* insere um controlo que vai estar ligado ao subformulário. Podemos ver a forma como são ligados acedendo às propriedades do subformulário, mostradas na figura seguinte.

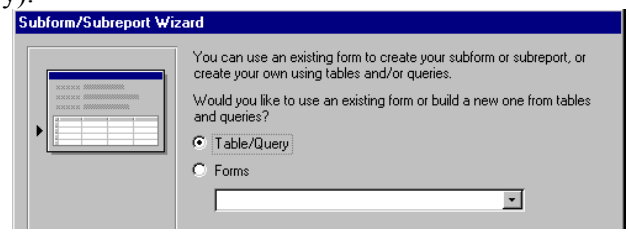


O campo Source Object indica o formulário a incluir, isto é, a mostrar como subformulário. Em seguida é preciso determinar como esse formulário será ligado ao formulário principal, isto é, que campos de um e outro permitem ligá-los. Neste caso ligámos o campo número do formulário principal (Link Master Fields) ao campo com o mesmo nome do subformulário (Link Child Fields).

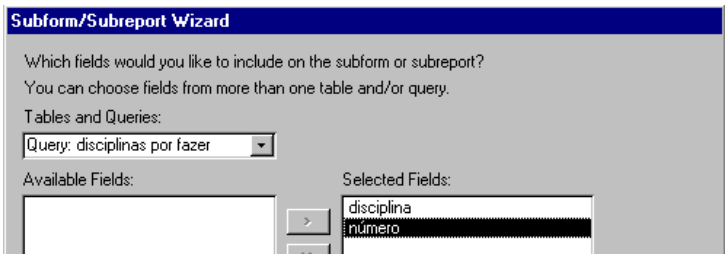
5.9 Controlo subform

Outra forma de acrescentar subformulários a formulários já existentes é usando o *wizard* do controlo subform.

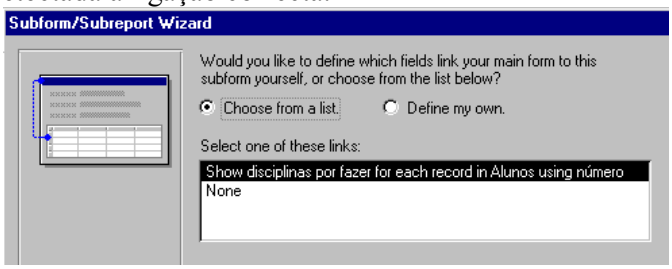
Supondo a existência de uma *query* que nos indique as disciplinas que faltam fazer a cada aluno, chamada disciplinas por fazer, para juntarmos essa informação ao formulário criado atrás devemos primeiro entrar na *design view* do formulário principal. Em seguida criar um controlo do tipo Subform/Subreport como explicado atrás para os outros controlos. Em seguida podemos seleccionar se queremos acrescentar um formulário já existente (Forms) ou criar um formulário novo a partir de uma tabela/*query* (Table/Query).



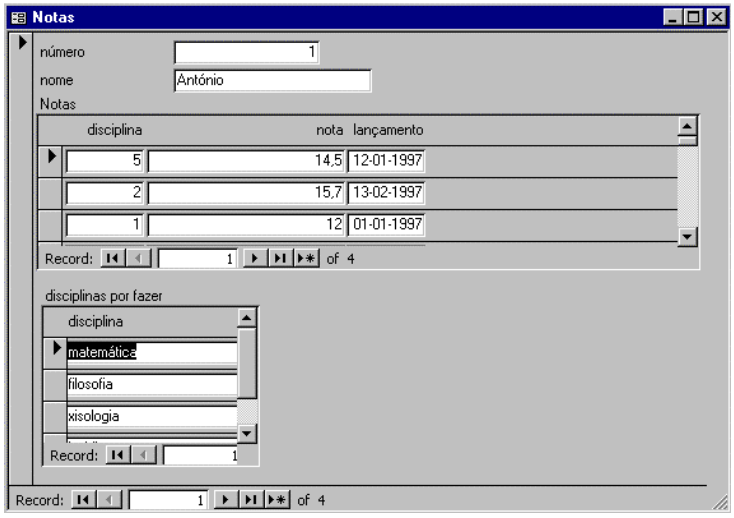
Depois seleccionamos os campos que desejamos incluir no subformulário. Note-se que é necessário incluir o campo número que permitirá a seguir fazer a ligação ao formulário principal.



É então apresentada uma caixa de diálogo onde nos é pedido para identificar a ligação entre a tabela/*query* do formulário principal e a escolhida para o subformulário. No caso do *Access* detectar a existência de um relacionamento entre elas, apresenta uma sugestão (Choose from list). Caso contrário, será preciso indicar (Define my own) um ou mais campos no formulário principal (Form/report fields) que farão a ligação com campos correspondentes indicados para o subformulário (Subform/subreport fields). Neste caso foi detectada a ligação correcta.



Finalmente dá-se um nome ao subformulário e o resultado é apresentado. É conveniente alterar a apresentação do formulário de forma a que o número do aluno não apareça repetido.


Notas
número
1
nome
António
Notas
disciplina
nota
lançamento
5
14,5
12-01-1997
2
15,7
13-02-1997
1
12
01-01-1997
Record:
1
of 4
disciplinas por fazer
disciplina
matemática
filosofia
xisologia
Record:
1
Record:
1
of 4

ANEXOS

6 ANEXO A

6.1 Entidades e Relacionamentos

Este método representa um processo do geral para o específico.
Para problemas complexos, definidos por muitos atributos, os estados iniciais das relações, quando abordadas pelo método da normalização, são extensos e com dependências complexas. Torna-se fastidioso e sujeito a enganos, trabalhar com as relações não normalizadas.

6.1.1 Entidades

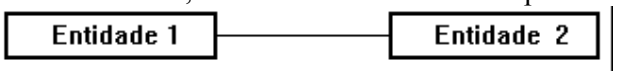
Um outro processo de obter um modelo de dados que satisfaz os objectivos inicialmente definidos, é a metodologia "Entidades e Relacionamentos". É uma metodologia que parte dos conceitos abstractos em jogo e os vai especializando. Por esse motivo, os passos intermédios são mais simples, envolvendo menos informação. O resultado final, é um conjunto de relações normalizado.
Os conceitos essenciais, nesta metodologia, são o conceito de "Entidade" e o de "Relacionamentos" entre as entidades. Uma entidade é um objecto abstracto considerado importante para o sistema. No caso em estudo, poderemos considerar que a situação envolve Leitores e Livros cuja interacção é registada sob a forma de uma Requisição. Estas serão as entidades importantes para o modelo de dados. É usual representa-las como:

Livros	Leitores	Requisição

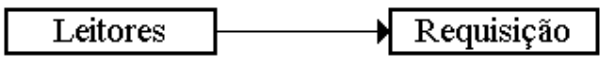
Cada Entidade é descrita por um conjunto de propriedades, designadas por atributos:
livros(colocação, título, autor, editor, edição, ano)
leitores(bilhete_identidade, nome, localidade, telefone, assinatura)
requisição(número, data)
Os atributos sublinhados constituem as chaves das relações.

6.1.2 Relacionamentos entre Entidades

As interacções entre as entidades são descritas pelos seus relacionamentos. No caso da biblioteca, uma análise dos relacionamentos entre as entidades, revela:
 Um leitor pode efectuar várias requisições, mas cada requisição só pode ser feita por um único leitor.
 Numa requisição, podem ser requisitados vários livros, e o mesmo livro pode ser requisitado várias vezes (em momentos diferentes).
 Toda a interacção entre leitores e livros tem de ser efectuada sob a forma de uma requisição: não existe um relacionamento directo entre Livro e Leitor. O modelo, considera três tipos de relacionamentos:
 Relacionamentos 1:1 - Quando a cada valor da chave de uma entidade, corresponde um e um só valor da chave da outra entidade. No caso da biblioteca, não há um relacionamento entre entidades, deste tipo. Deixamos ao leitor, como exercício, o encontrar em situações do seu interesse, relacionamentos deste tipo. Graficamente, relacionamentos 1:1 são representados por:

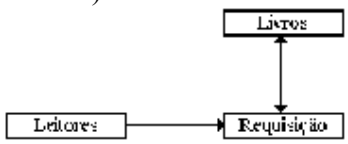


 Relacionamentos 1:N - Quando a cada valor da chave de uma entidade correspondem vários valores da chave da outra. É o caso do relacionamento Leitores - Requisições, que é representada por:

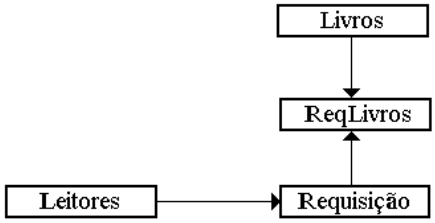


 Relacionamentos N:M - Quando a cada valor da chave de uma entidade correspondem vários valores da chave da outra entidade, e vice-versa. É o caso do relacionamento Livros - Requisição, que é representada por:

Diagrama de Entidades e Relacionamentos
Graficamente, os diversos relacionamentos entre as entidades, são representados pelo diagrama de Entidades - Relacionamentos (diagrama E-R):



Os relacionamentos N:M são difíceis de implementar na maior parte dos SGBD, porque requerem uma relação simétrica, isto é de Entidade1 Entidade2 e de Entidade2 Entidade1.
No entanto é sempre possível desdobrar um relacionamento N:M em dois relacionamentos 1:N e 1:M, criando uma *entidade de ligação*.
Por exemplo, o relacionamento N:M, Livros Requisição pode ser desdobrado em dois relacionamentos: Livros ReqLivros e Requisição ReqLivros, sendo ReqLivros a entidade de ligação.
A versão final do diagrama E-R, fica:



Ao definir os atributos de cada relação, e para garantir a consistência da informação, é necessário "importar" chaves. Uma requisição, por exemplo, é caracterizada por um número e uma data. Desta forma, não poderíamos saber quem efectuou a requisição. Mas, como mantém uma relação N:1 com Leitores, vai herdar a chave de leitores: ou seja, a chave de Leitores passa a fazer parte dos atributos de Requisição, tornando possível determinar quem efectuou uma dada requisição.
Uma entidade importa a chave das entidades com as quais tem uma relação N:1
Utilizando esta regra, a relação Requisição importa a chave de Leitores e a relação ReqLivros importa as chaves de Livros e de Requisição. Os atributos de cada relação, serão:
livros(colocação, título, autor, editor, edição, ano)
leitores(bilhete_identidade, nome, localidade, telefone, assinatura)
requisição(número, bilhete_identidade, data)
reqlivro(número, colocação, data_devolução)

Esta solução não contempla o problema da temática de cada livro. É proposto, ao leitor, efectuar as alterações necessárias ao diagrama E-R, assim como ás relações, de forma a englobar esta situação.

Implementação do Modelo
Geralmente o estudo do modelo de dados, tem como objectivo final, uma implementação, utilizando um Sistema de Gestão de Bases de Dados. Esta fase envolve passar de uma definição lógica dos dados para uma descrição física. Dois aspectos a considerar:

- . Definição das Tabelas - a regra geral é de que cada relação dá origem a uma tabela.

- . Definição dos campos da Tabela - a regra geral é de que os campos de cada tabela são os atributos da entidade respectiva, eventualmente acrescidos das chaves importadas.

De uma forma geral, pretendem-se implementações que, *preservando as propriedades do modelo*, possibilitem um bom desempenho nas operações de manutenção da Base de Dados e nos processos de transacção.

Uma forma de melhorar o desempenho, pode ser obtida minorando o número de Tabelas. Um exemplo típico é o caso dos relacionamentos 1:1. Nesta situação, é normalmente decidida a junção dos atributos das duas entidades, assim relacionadas, numa única Tabela.

Outra forma de melhorar o desempenho, consiste em optimizar o acesso aos registos: utilizar acesso directo (obtido através da indexação) por oposição ao acesso sequencial. A chave de uma relação é usualmente utilizada como índice para a Tabela respectiva. No caso da importação de chaves, é normal que a chave importada venha a fazer parte da chave da Tabela. É o caso das entidades de ligação, cuja chave é sempre a conjunção das duas chaves importadas. No entanto, nem todas as situações de importação de chaves verificam esta regra. Poderão existir outras conjunções de atributos, que por si só, poderão ser chave da relação, sem necessidade de incorporação da chave importada: é o caso da entidade ***requisição*** (no exemplo da biblioteca), que importa a chave da entidade ***leitores***, no entanto, esta não faz parte da chave de ***requisição*.**

No caso da biblioteca, e supondo que utilizamos o *SGBD Access*, o conjunto de Tabelas seriam:

Tabela: **Livros**, chave Colocação

Campo	**Tipo**	**Tamanho**
Colocação	Text	6
Título	Text	30
Autor	Text	25
Editor	Text	25
Edicao	Text	10
Ano	Integer	

Tabela: **Leitor**, chave BI

Campo	**Tipo**	**Tamanho**
BI	Integer	
Nome	Text	30
Localidade	Text	25
Telefone	Text	8

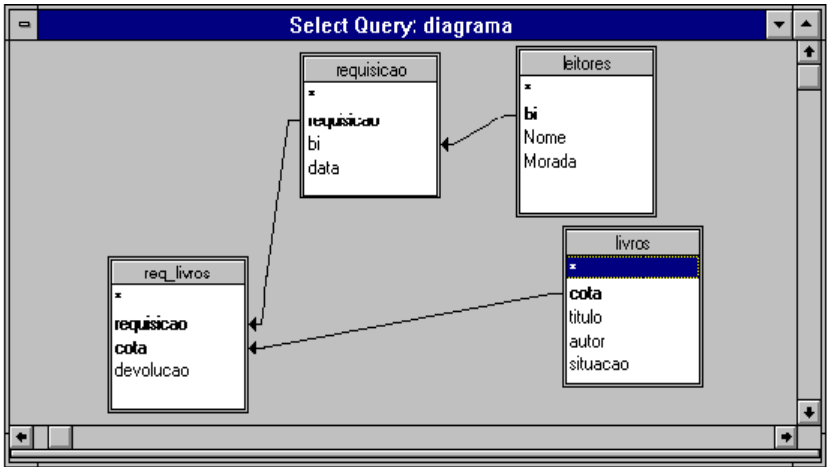
Tabela: **Requisição**, chave ReqNr

Campo	**Tipo**	**Tamanho**
ReqNr	Integer	
BI	Integer	
Data	Date	

Tabela: **ReqLivros**

Campo	**Tipo**	**Tamanho**
ReqNr	Integer	
Colocação	Text	6
DataDevol	Date	

Desta forma, a informação fica distribuída por diferentes Tabelas. Para saber o nome da pessoa que fez uma determinada requisição, é necessário consultar, pelo menos, duas Tabelas. É sempre possível reconstituir a informação original, relacionando as tabelas. A figura ilustra o diagrama de relacionamentos entre as diferentes tabelas:


Select Query: diagrama
requisicao
*
requisicao
bi
data
leitores
*
bi
Nome
Morada
req_livros
*
requisicao
cota
devolucao
livros
*
cota
titulo
autor
situacao

7 ANEXO B

Tabelas

Todos os dados que são armazenados de forma permanente pelo Access, são guardados em **tabelas**. Cada tabela armazena dados sobre um tipo de coisa, pessoa ou relação entre coisas/pessoas. Numa tabela pode-se, por exemplo, guardar informação sobre alunos, disciplinas ou notas (que podem ser vistas como relações entre alunos e disciplinas). As tabelas estão organizadas em linhas e colunas. Em cada linha guarda-se dados sobre um item, isto é, sobre um aluno, disciplina ou nota, respectivamente para os exemplos dados. Cada coluna representa uma determinada característica que está associada a todos os itens do tipo que a tabela representa. Por exemplo, numa tabela de alunos deverá existir uma coluna para o nome e outra para a data de nascimento. Em linguagem de bases de dados (BD), uma linha é um **registo** (*record*) e uma coluna é um **campo** (*field*). É possível ter um número potencialmente ilimitado de registos em cada tabela, mas o número de campos é fixo e limitado.
Cada campo de uma tabela tem um **tipo** associado que limita o tipo de dados que lá podem ser guardados. O campo "nome" da tabela de alunos será do tipo texto e o campo "data de nascimento" será do tipo data. Assim, no campo "nome" poderão ser introduzidos caracteres alfanuméricos enquanto no campo "data de nascimento" apenas podem ser guardadas datas. Ao conjunto de campos que compõe uma tabela, chama-se a **estrutura** da tabela. O tipo determina também as operações que podem ser efectuadas com valores do campo. Por exemplo, não é possível fazer uma soma que envolva um campo do tipo texto.
A informação inserida numa tabela é **persistente**. Isto é, cada registo criado numa tabela é imediatamente guardado no ficheiro da BD.

Criação de Tabelas

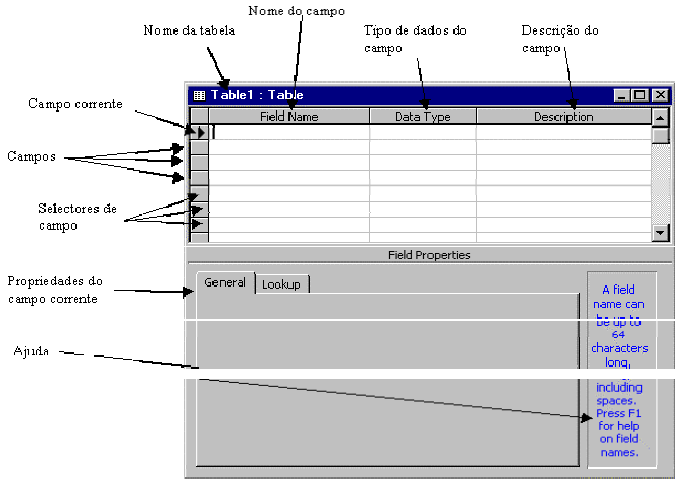
Para criar uma tabela é necessário seleccionar o separador Tables na janela de BD, clicar no botão New. Aparece então uma caixa de diálogo que permite seleccionar a forma como a tabela será criada:
As opções disponíveis são:

- **Datasheet View [Vista de folha de dados]**: o utilizador introduz dados numa matriz semelhante a uma folha de cálculo e o Access procura inferir os tipos dos campos.
- **Design View [Vista de estrutura]**: definição explícita da estrutura da tabela.
- **Table Wizard [Assistente de Tabelas]**: o assistente disponibiliza um conjunto de campos pré-definidos e dividido em categorias com os quais o utilizador pode constituir a sua tabela.
- **Import Table [Importação de Tabelas]**: construção de tabelas a partir de ficheiros já existentes, por exemplo, documentos Excel ou ficheiros de texto.
- **Link Table [Ligação de Tabelas]**: possibilita que o Access tenha acesso a uma tabela de um aplicação/SGBD diferente sem ser preciso duplicar a tabela nem que os utilizadores da aplicação/SGBD original percam o acesso aos mesmos dados.

Embora tanto a vista de folha de dados como o assistente possam parecer mais apelativos à primeira vista, a utilização dessas opções trazem algumas limitações e dificuldades para além de serem específicas do Access. Assim, será dado ênfase à vista de estrutura.
Exemplo: Para exemplificar a criação de uma tabela supõe-se que as pessoas com os dados que se seguem se inscreveram numa escola:

Carolina Santos	R. Cedofeita, 14	Porto	11-12-1979
Maria Oliveira	R. Igreja, 3	Lisboa	23-3-1978
António Silva	Tr. Povo, 8	Moura	12-5-1979
Paulo Castro	R. Flores, 50	Porto	23-4-1979
Ana Oliveira	Pç. Município, 1	Moura	2-2-1979

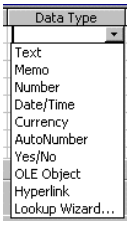
Como se pode ver, a organização dos dados permite identificar facilmente a estrutura da tabela: os campos necessários são nome, morada, localidade e data de nascimento. Note-se que nem sempre é tão fácil perceber a estrutura da tabela a partir dos dados.
Depois de escolher a opção Design View na caixa de diálogo anterior, é apresentado a seguinte janela:



7.1.1 Definição de Campos

Como explicado atrás, um campo é definido por um nome (Field Name) e um tipo de dados (Data Type). O Access possibilita ainda a introdução de uma pequena descrição do conteúdo do campo (Description).
O nome identifica o campo e possibilita aceder à informação nele contida. Embora tal possa não ser evidente à partida, a escolha dos nomes dos campos é muito importante para facilitar a utilização das tabelas. Será mais difícil perceber a que diz respeito o conteúdo de um campo se o nome for DN do que se for Data de Nascimento.

A escolha do tipo de dados é feita utilizando o menu:



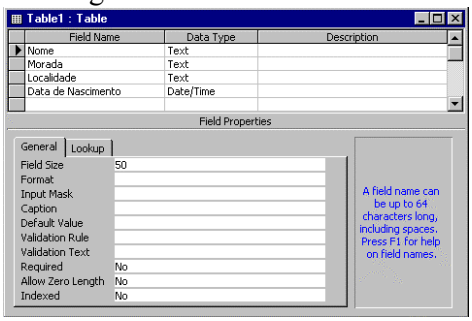
O tipo de dados determina quer o tipo de informação que o campo pode armazenar, quer o espaço (em bytes) que a informação ocupa. Os tipos disponíveis no Access são:

- **Text [Texto]**: texto (até 255 caracteres alfanuméricos).
- **Memo**: texto (até 64.000 caracteres alfanuméricos).

- **Number [Número]** : valores numéricos.
- **Date/Time [Data/Hora]**: datas e horas.
- **Currency [Moeda]**: valores monetários.
- **AutoNumber [Numeração Automática]**: valores inteiros atribuídos automaticamente pelo Access (anteriormente ao Access 97 este tipo era chamado Counter)
- **Yes/No [Sim/Não]**: valores lógicos (Booleanos),
- **OLE object [Objecto OLE]**: inclusão ou ligação de desenhos, imagens e gráficos (até 128 MB).
- **Hyperlink [Hiperligação]**: apontador para outro ficheiro ou URL (*Uniform Resource Locator*), por exemplo o endereço de uma página *World Wide Web*.

A opção **Lookup Wizard… [Assistente de Pesquisa…]** não representa um tipo. Na realidade possibilita a definição de um campo e dos valores que pode tomar, ou manualmente ou a partir de outro campo, normalmente de outra tabela.
Uma vantagem de atribuir tipos aos campos é limitar os erros que os utilizadores podem fazer ao introduzir dados. Por exemplo, se um campo estiver definido como sendo do tipo Number não é possível introduzir letras.
A escolha do tipo de dados está relacionada com as operações que se pretende efectuar nesse campo. Quando fôr necessário efectuar operações aritméticas com os valores de um campo, então será melhor utilizar o tipo Number ou Currency (este último no caso de se estar a lidar com valores monetários). Por exemplo, para que seja possível calcular médias, o campo que guardar notas dos alunos deverá ser do tipo Number. Por outro lado, para guardar informação sobre nomes de pessoas convém usar o tipo Text. Nalguns casos a escolha poderá não ser tão imediata. Um exemplo comum são os campos onde se guarda números de telefone. Dado que não faz sentido efectuar operações aritméticas com esse tipo de informação podemos usar o tipo Text. Com este tipo de dados é possível efectuar a operação de concatenação, por exemplo, para acrescentar o indicativo da rede ao início do número. Por outro lado, se fôr usado o tipo Number, evita-se que o utilizador possa introduzir letras, dado que em Portugal os número de telefone são constituídos por algarismos.

Exemplo: Todos os campos da tabela de alunos são do tipo Text excepto Data de Nascimento que é do tipo Date/Time, como se pode ver na figura:



Como os nomes dos campos são óbvios torna-se desnecessário introduzir qualquer descrição. Dado que o campo corrente está definido, já é possível ver na secção de baixo da janela as suas propriedades, que serão explicadas a seguir.
Antes de prosseguir, grava-se a tabela (File, Save). O Access pede um nome que poderá ser Alunos.

7.1.2 Definição das Propriedades dos Campos

Como se pode ver na figura acima, existe um conjunto de propriedades associado a cada campo. Essas propriedades permitem refinar as características desse campo. Para cada tipo de dados as propriedades variam. Uma utilização adequada destas propriedades diminui a possibilidade de introdução de dados errados e facilita a utilização da BD de forma significativa.

Em seguida são descritas as mais importantes, todas referentes ao separador General, indicando os tipos de dados a que dizem respeito:

FieldSize [Tamanho de Campo]

- Text: Número de caracteres máximo. O tamanho definido pelo Access quando se cria um campo tipo texto é geralmente demasiado grande para as necessidades do utilizador.

Exemplo: O tamanho dos campos Nome e Localidade da tabela Alunos será, respectivamente 30 e 20, enquanto o do campo Morada pode ficar nos 50 que o Access preenche automaticamente.

- Number: Byte, Integer, Long Integer e ReplicationID (ver AutoNumber) para números inteiros, e Single ou Double para números reais. O tamanho de um número permite definir a gama de valores que são representáveis.
- AutoNumber: Long Integer que é equivalente ao mesmo tamanho para Number ou ReplicationID, sendo este último necessário apenas para BDs partilhadas, assunto que está fora do âmbito deste texto.

Format [Formatar] (todos os tipos excepto OLE Object) Os formatos possibilitam definir a maneira como o valor do campo vai ser visualizado e imprimido. O utilizador pode definir os seus próprios formatos ou usar formatos pré-definidos quando disponíveis:

- Number: General Number, Currency, Fixed, Standard, Percent e Scientific. O formato Fixed permite representar valores com um número pré-determinado de casas decimais. O formato Scientific permite representar números reais.

Exemplo: O número "1200" no formato Scientific é representado como 1.2E+3 (isto é, o número 1.2 x 10^3).

- Date/Time: Os formatos para datas são General Date, Long Date, Medium Date e Short Date. Os formatos para tempo são: Long Time, Medium Time e Short Time.

Exemplo: A data "26 de Fevereiro 1996" no formato Medium Date aparece como 26-February-96. Se fôr utilizado Short Date, a data terá a forma 2/26/96 (repare que o mês aparece em primeiro lugar).

Exemplo: Utilizando o formato Short Time, a hora "17h30" seria apresentada como 17:30.

Exemplo: O formato do campo Data de Nascimento da tabela Alunos pode ser Medium Date dado que não é necessário saber a hora de nascimento.

Input Mask [Máscara de Edição] (todos os tipos excepto Memo, AutoNumber, Yes/No e OLE Object) As máscaras de edição formatam o campo no momento da introdução ou alteração de dados de forma a tornar esse processo mais simples para o utilizador.

Exemplo: A Input Mask do campo Data de Nascimento deve corresponder ao formato definido na propriedade Format, ou seja, Medium Date. Tal pode ser feito facilmente usando o assistente.

Default Value [Valor predefinido] (todos excepto AutoNumber e OLE Object) Definição de um valor a ser preenchido automaticamente pelo Access quando é criado um registo novo.

Exemplo: Assumindo que a maior parte dos alunos são do Porto, o Default Value do campo Localidade será "Porto".

Validation Rule e **Validation Text [Regra de Validação** e **Texto de Validação]** (todos os tipos excepto AutoNumber e OLE Object) Estas propriedades permitem, respectivamente, definir uma expressão que tem que ser verificada pelos valores introduzidos no campo e o texto a mostrar na caixa de diálogo apresentada ao utilizador no caso da regra não se verificar.

Exemplo:

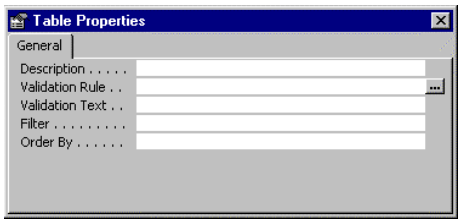
Validation Rule	Validation Text
>=0	Digite um Número Positivo
>=0 AND =<20	Nota compreendida entre 0 e 20
0 OR >=100	Valor tem que ser zero ou maior que 100
Like "K???"	Digite 4 caracteres, começando por K
<=#1/1/95#	Digite data anterior a 1995
>=Date()	Digite data não anterior à de hoje

Alguns comentários relativos ao exemplo anterior:

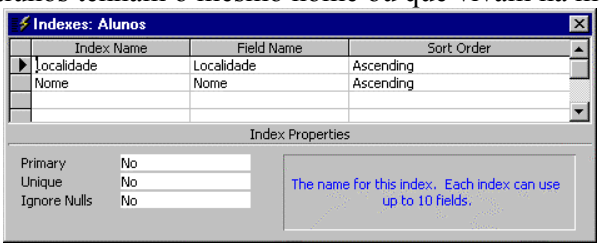
- Os textos são dados entre aspas.
- É possível usar caracteres de substituição (*wild cards*) nas regras de validação de campos de texto. ? representa "um caracter qualquer" e * representa "0, 1 ou mais caracteres quaisquer".
- As datas são representadas entre sinal de cardinalidade (#).
- Pode-se usar funções, como no caso do último exemplo em que foi usada a função Date que devolve a data de hoje.

Exemplo: Dado que os alunos têm que ter pelo menos 17 anos para entrar na faculdade a regra de validação do campo Data de Nascimento será: Year([Data de Nascimento]) <= Year(Date()) - 16. A função Year extrai o ano de uma data que recebe como argumento.

Não é permitido fazer referência a outros campos nas expressões usadas na Validation Rule de um campo. Para criar regras de validação que envolvam mais do que um campo é necessário recorrer à Validation Rule nas propriedades da tabela:



Required [Necessário] (todos menos AutoNumber) Definir obrigatoriedade de preencher o campo.
Exemplo: Na tabela Alunos é necessário que todos os campos sejam preenchidos, por isso a propriedade Required é activada em todos.
Indexed [Indexado] (todos menos Memo, OLE Object e HyperLink) Definir um índice para aumentar a velocidade de procura de registos por valores do campo respectivo. Este aumento de velocidade de acesso tem custos, nomeadamente em termos de espaço ocupado pelos índices e da velocidade na criação de registos, por causa da necessidade de actualizar todos os índices para cada novo registo. Por isto, a escolha dos campos a indexar tem que ser criteriosa, seleccionando apenas aqueles pelos quais se pretende efectuar procuras.
Há 3 valores possíveis para a propriedade Indexed: No, não cria índice;Yes (Duplicates OK), cria um índice e permite duplicados, isto é, que 2 registos tenham o mesmo valor para o campo em questão; Yes (No Duplicates), cria um índice mas não permite valores duplicados.
Exemplo: Na tabela de alunos será necessário, por exemplo, procurar registos por Nome e por Localidade. A propriedade Indexed é activada nestes campos usando o valor Yes (Duplicates OK) já que é possível que vários alunos tenham o mesmo nome ou que vivam na mesma localidade.

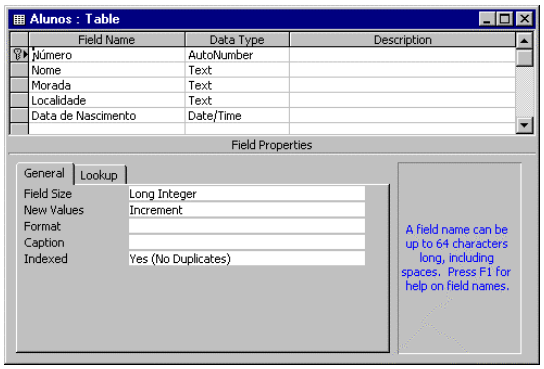


Por vezes existem tabelas em que são frequentes procuras com critérios que envolvem mais do que um campo. Para construir um índice com vários campos é preciso abrir a janela de índices (View, Indexes).
Na figura pode-se ver a janela de índices para a tabela Alunos. Podemos ver os índices para os campos Nome e Localidade criados através da propriedade Indexed desses campos. Todos os índices têm um nome (Index Name) que, no caso de envolver apenas um campo, é o nome desse campo. É também

possível definir ordem crescente ou decrescente na coluna Sort Order. Criar um índice com vários campos é semelhante a criar vários índices seguidos só com um campo, apenas com a diferença de que o nome do índice é definido na primeira linha e deixado em branco nas linhas dos outros campos.

7.1.3 Definição da Chave

Em princípio, todos os registos de uma tabela serão diferentes entre si. Por esse motivo deverá ser possível identificar um campo (ou conjunto de campos) tal que conhecido o seu valor, é possível identificar de forma única o registo correspondente. Esse campo (ou conjunto de campos) é designado por **chave** (*key*). Uma tabela pode ter mais do que uma chave. Nesse caso escolhe-se uma e designa-se por **chave primária** (*primary key*).
Exemplo: Numa BD que inclua campos para o número de Bilhete de Identidade e o número de contribuinte (NIF), qualquer um desses campos serve como chave. Dependendo da aplicação, escolhe-se um deles para ser a chave primária.
É recomendável a definição de chaves para todas as tabelas que se cria: vão ter um papel importante no estabelecer de relacionamentos entre tabelas, possibilitando aceder a informação distribuída por diferentes tabelas. Por vezes nem mesmo todos os campos permitem identificar o registo (isto é, há registos duplicados) ou não existe nenhuma chave que faça sentido na aplicação em questão. Nesse caso, acrescenta-se um campo com esse fim que, no Access, tem normalmente o tipo de dados AutoNumber.
Para definir a chave da tabela, deverá seleccionar o(s) campo(s) chave (clicando no selector de campo respectivo, usando a tecla CTRL no caso de mais do que um campo) e executar o comando File, Primary Key.



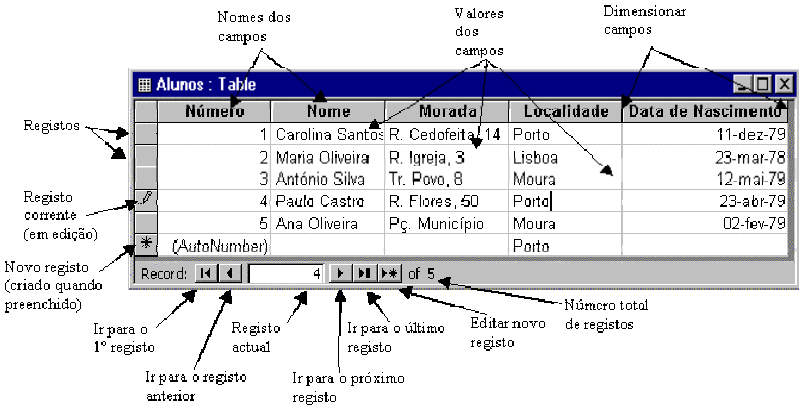
Exemplo: Na tabela Alunos, dado que pode haver vários alunos com o mesmo nome, mesma morada, mesma localidade e mesma data de nascimento, nenhum dos campos existentes serve como chave. Assim, acrescentámos um campo Número do tipo AutoNumber que definimos como chave primária. A tabela ficará com o seguinte aspecto:

É preferível não usar como chave campos de texto com espaço para muitos caracteres, como, por exemplo, campos que guardam nomes porque é frequente os utilizadores introduzirem erros de difícil detecção neste tipo de campos (espaços a mais ou a menos, falta de acentuação ou maiúsculas em vez de minúsculas e vice-versa). No caso de não haver nenhuma boa alternativa, será melhor criar um campo código do tipo AutoNumber.

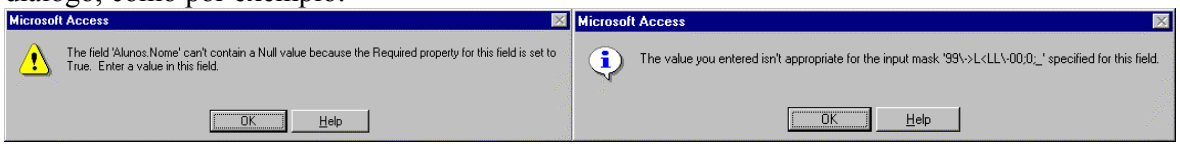
7.1.4 Edição de Dados

Definida a estrutura da tabela, esta está apta a guardar informação. Pode-se editar (inserir, eliminar registos ou alterar campos) directamente na tabela, apesar de ser mais apropriado efectuar estas operações via formulários.
Para abrir a tabela executa-se o comando File, Open sendo necessário que a tabela Alunos esteja seleccionada na janela da BD. A tabela é apresentada com o aspecto de uma folha de cálculo, como se pode ver na figura. Os campos estão dispostos em colunas e os registos em linhas. Os registos estão

ordenados pelo campo chave. O último registo da tabela, marcado com * no selector de registos, é utilizado para inserir novos registos.
Exemplo: Depois de criada, insere-se os dados na tabela Alunos:



Note-se que o registo actual indica o número do registo, o que nesta tabela é por acaso o mesmo valor do campo Número.
Quando edita dados num registo, toda a informação indicada na criação da tabela é utilizada: os campos são preenchidos inicialmente com os valores pré-definidos, é feita a validação do tipo de dados e aplicadas as regras de validação definidas. Para o utilizador se aperceber de quando está a alterar um registo, o ícone de registo corrente é substituído por um lápis. As situações de erro dão origem a caixas de diálogo, como por exemplo:



Para eliminar um registo é necessário selecioná-lo (clicando no selector de registo) e carregar na tecla Delete ou colocar o cursor em qualquer campo do registo e executar o comando Edit, Delete.

7.1.5 Alteração da Estrutura das Tabelas

Depois de definir a tabela, pode em qualquer momento eliminar e alterar campos existentes, inserir campos novos ou alterar a ordem pela qual os campos estão dispostos. Para efectuar estas operações são utilizados os processos usuais no ambiente gráfico do Windows: o rato, o menu Edit e a barra de ferramentas.