Computer Science

COMPSCI 111 SC - Lecture 24/25

September 2003

Storing and Retrieving Information

Databases

Introduction

Long before computers existed we have been creating "*databases*" and using them for keeping important information. People maintain address books, dentists and doctors maintain patient records, libraries have kept catalogues of their books and automobile dealers keep track of who their customers are and when their cars are serviced. A database is a collection of data about a particular subject. Information is stored so it can be easily retrieved. With the advent of computers, storing and retrieving large amounts of information has become easier. If you want to find a customer by surname, it may not take too long to search through filing cabinets and pull out the correct record. If instead you want to find all of the people who have purchased red cars since 1997 this would be a difficult manual task. But it is easy to do with an electronic database.

Databases

Databases are collections of data about a particular subject. We can think of this as an electronic filing cabinet. The user of the database is given a variety of facilities to perform on the information contained in the database. They may:

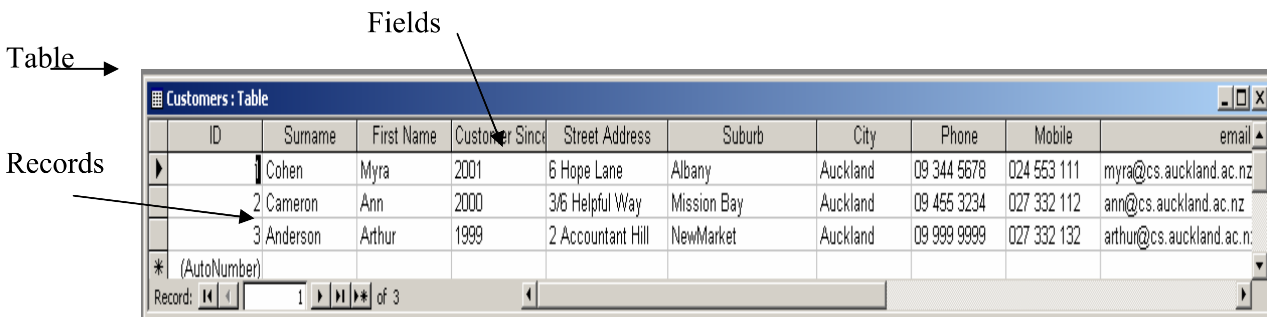
- Add new files to the database
- Insert new files into the database
- Retrieve existing files
- Update data in existing files
- Delete data from existing files
- Remove existing files permanently from the database

Why do we use a database?

- **Compact**: We no longer need to have rooms full of filing cabinets. The electronic version of this information can be stored on computer disks in a fraction of the space.
- **Speed:** A computer can retrieve and update information faster than a human
- **Accuracy**: we can apply integrity checks to our electronic database to help insure we are entering/storing accurate information.
- **Standards**: We can enforce standards in representing information
- **Security**: We can apply security restrictions to all or parts of the information contained in the database.

A database is a collection of **records**. A record holds a single unique fact about an object in the database. A record consists of many **fields**. A field is one piece of information making up the fact. For instance we might have a record for Myra Cohen purchasing a red car. Myra Cohen, the colour red, the type of car are all fields that make up this fact.

Below is a database table with 3 records.



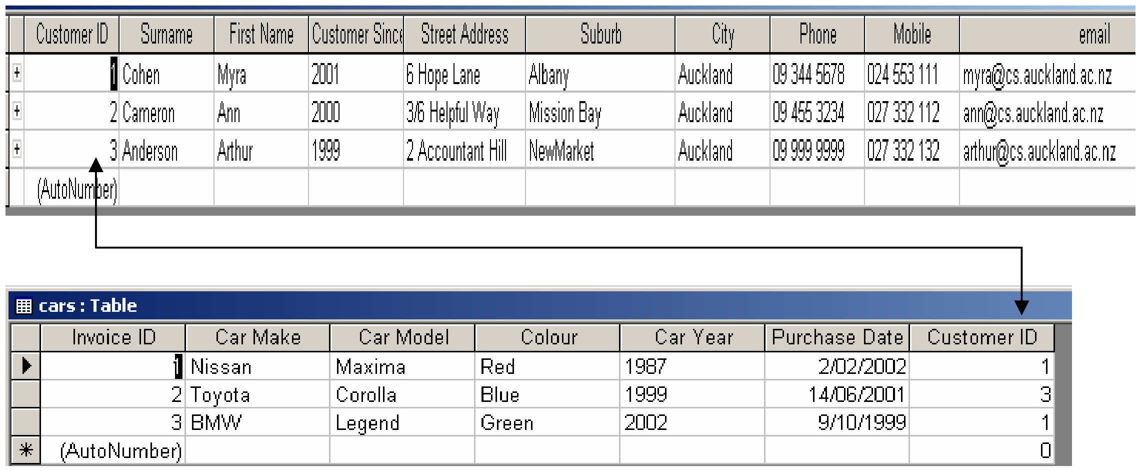
ID	Surname	First Name	Customer Sinc	Street Address	Suburb	City	Phone	Mobile	email
1	Cohen	Myra	2001	6 Hope Lane	Albany	Auckland	09 344 5678	024 553 111	myra@cs.auckland.ac.nz
2	Cameron	Ann	2000	3/6 Helpful Way	Mission Bay	Auckland	09 455 3234	027 332 112	ann@cs.auckland.ac.nz
3	Anderson	Arthur	1999	2 Accountant Hill	NewMarket	Auckland	09 999 9999	027 332 132	arthur@cs.auckland.ac.n
(AutoNumber)									

This Database example has **3 records**. Each record has **10 fields**.

Relational Databases
There are many ways to store information in a database. The **relational database model** was introduced in the 1970's by E. F. Codd. This model allows the user to see a conceptual (**logical) view** of the data and to ignore the physical interpretation. We can relate information stored in separate tables by the logical relationships among them. The way the database software physically stores the data on a computer disk system is called the **internal view**. The internal view differs in different database management programs. We will not concern ourselves with these details. The relational model is the most common database model used today.

In a relational database we have:

Relations (Tables):
Each table has **columns** (fields) and **rows** (records)

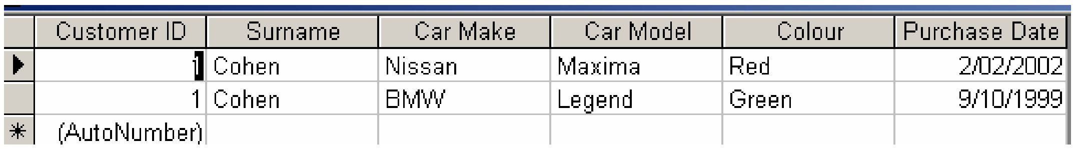


Customer ID	Surname	First Name	Customer Since	Street Address	Suburb	City	Phone	Mobile	email
1	Cohen	Myra	2001	6 Hope Lane	Albany	Auckland	09 344 5678	024 553 111	myra@cs.auckland.ac.nz
2	Cameron	Ann	2000	3/6 Helpful Way	Mission Bay	Auckland	09 455 3234	027 332 112	ann@cs.auckland.ac.nz
3	Anderson	Arthur	1999	2 Accountant Hill	NewMarket	Auckland	09 999 9999	027 332 132	arthur@cs.auckland.ac.nz
(AutoNumber)									

Invoice ID	Car Make	Car Model	Colour	Car Year	Purchase Date	Customer ID
1	Nissan	Maxima	Red	1987	2/02/2002	1
2	Toyota	Corolla	Blue	1999	14/06/2001	3
3	BMW	Legend	Green	2002	9/10/1999	1
(AutoNumber)						0

We assign a data type to each field and define relationships among tables. Information between tables is linked via common fields. The two tables above are linked by the customer ID's.

We can see for instnace that Customer ID # 1 (Myra Cohen) has purchased 2 cars.



	Customer ID	Surname	Car Make	Car Model	Colour	Purchase Date
▶	1	Cohen	Nissan	Maxima	Red	2/02/2002
	1	Cohen	BMW	Legend	Green	9/10/1999
*	(AutoNumber)					

Each table has a **primary key.** This is a unique field or combination of fields for each record. The primary key cannot be null (empty). A **foreign key** is a field in a table that relates to the primary key of another table. This field must contain a value found in the primary key of the table to which it relates, or it must be null.
We use the primary key/foreign key relationship to define how our database tables are connected and to provide consistency of our data. We don't allow data to exist in a foreign key unless it has a matching value in our primary key. This is called **referential integrity**.

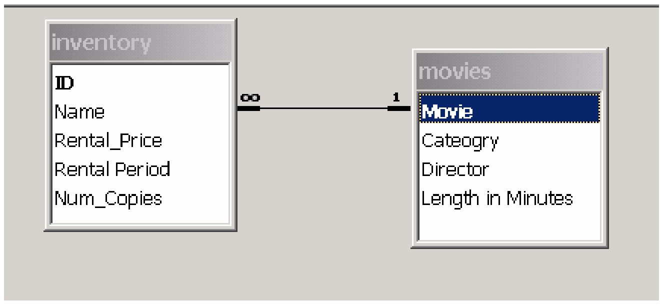


Table: Inventory – primary key= ID, foreign key=Name (relates to movies)
Table: Movies – primary key=Movie

The **Database management system (DBMS)** is the entire system used to store and retrieve the data. Microsoft Access is the DBMS we will use in this class.

Querying our Databases
In order to query our databases we must use a computer language. We will look at two of these here.

SQL
The most used computer language for querying databases is called **Structured Query Language** or SQL. This language is small and simple, yet allows us to perform all of the necessary operations to add, delete, update and modify tables in a database.

SELECT ***fieldname, fieldname, fieldname***
FROM ***tablename*****;**

When retrieving multiple columns, specify each field name. A comma must separate field names. The columns will display in the order you select them.

SELECT ***fieldname, fieldname*** **from** ***tablename***
ORDER BY ***fieldname*****;**

The Order By clause tells SQL you want the specified fields displayed in ascending order (ordered from A to Z, 1 to 100)

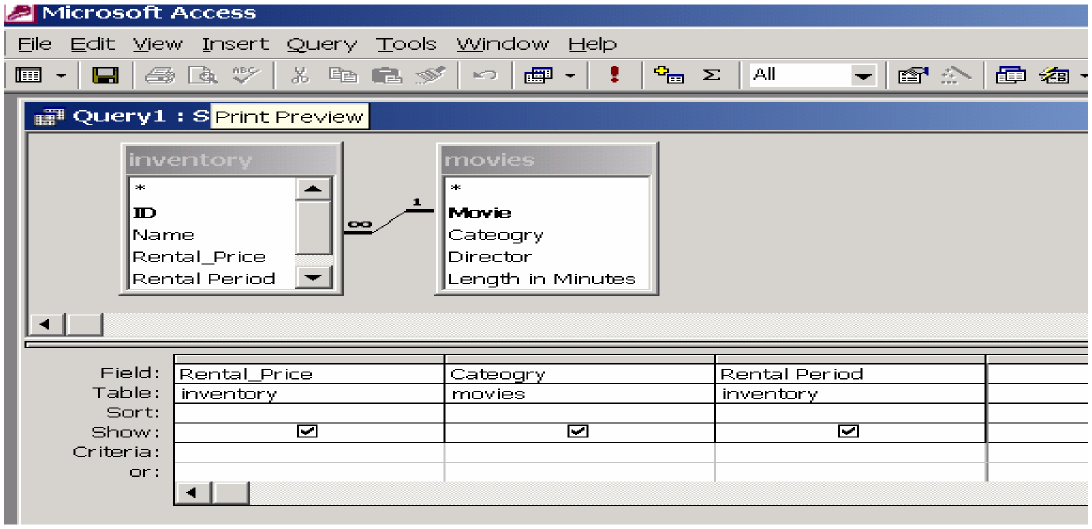
SELECT ***fieldname, fieldname,.. fieldname***
FROM ***tablename***
WHERE ***fieldname logical operator value***

Logical Operators	
=	Equal to
!= or <>	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
In	Equal to any item in a list
not in	Not equal to any item in a list
Between	Between two values, greater than or equal to one and less than or equal to the other
not between	Not between two values
begins with	Begins with specified value
Contains	Contains specified value
not contains	Does not contain specified value
is null	Is blank
is not null	Is not blank
Like	Like a specified pattern. % means any series of characters. _ means any single character.
not like	Not like a specified pattern. % means any series of characters. _ means many single character.

You can use any of the logical operators listed to constrain your field. In the Where clause, when referring to variables in character fields, you must enclose the values in single quotes. Variables that refer to numeric fields should ***not*** be enclosed in quotes.

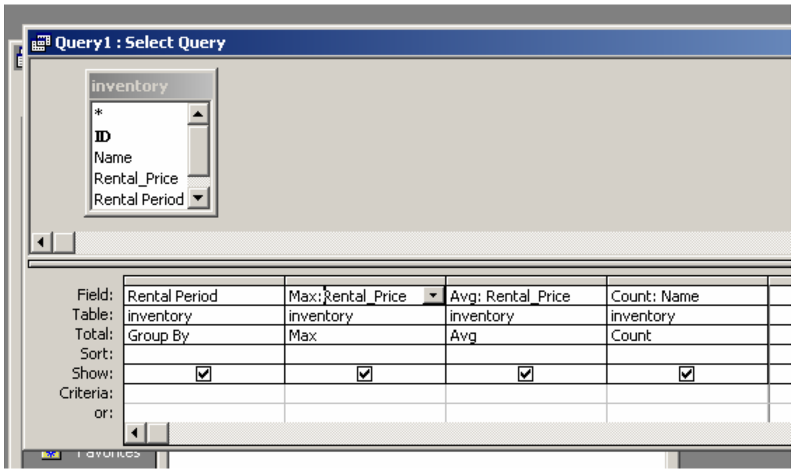
QBE

Many DBMS's such as MS Access provide graphical tools that allow us to create a query by moving fields onto a grid. This is called **Query By Example (QBE).** We can use this language to add constraints, sort, group and join tables of information.



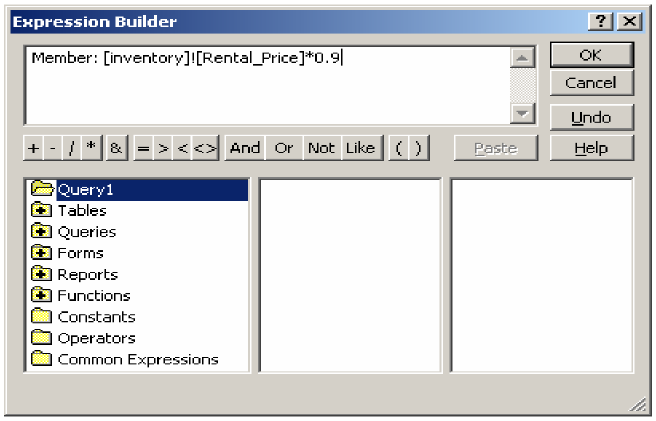
Field:	Rental_Price	Cateogry	Rental Period
Table:	inventory	movies	inventory
Sort:			
Show:	✓	✓	✓
Criteria:			
or:			

We can create aggregate (Totals) queries



Field:	Rental Period	Max: Rental_Price	Avg: Rental_Price	Count: Name
Table:	inventory	inventory	inventory	inventory
Total:	Group By	Max	Avg	Count
Sort:				
Show:	✓	✓	✓	✓
Criteria:				
or:				

Or create calculated fields



```
Member: [inventory]![Rental_Price]*0.9
```