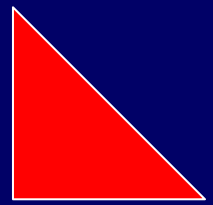
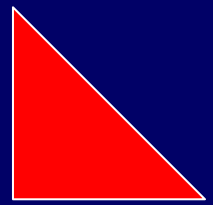


SQL 1

Unit 1.2

Database Models



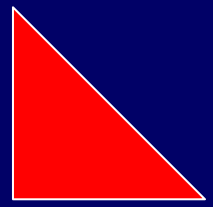
A data model comprises

- a data structure
- a set of integrity constraints
- operations associated with the data structure

Examples of data models include:

- hierarchic
- network
- relational

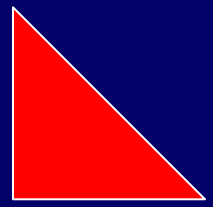
Relational Databases



The relational data model comprises:

- relational data structure
- relational integrity constraints
- relational algebra or equivalent (SQL)
  - SQL is an ISO language based on relational algebra
  - relational algebra is a mathematical formulation

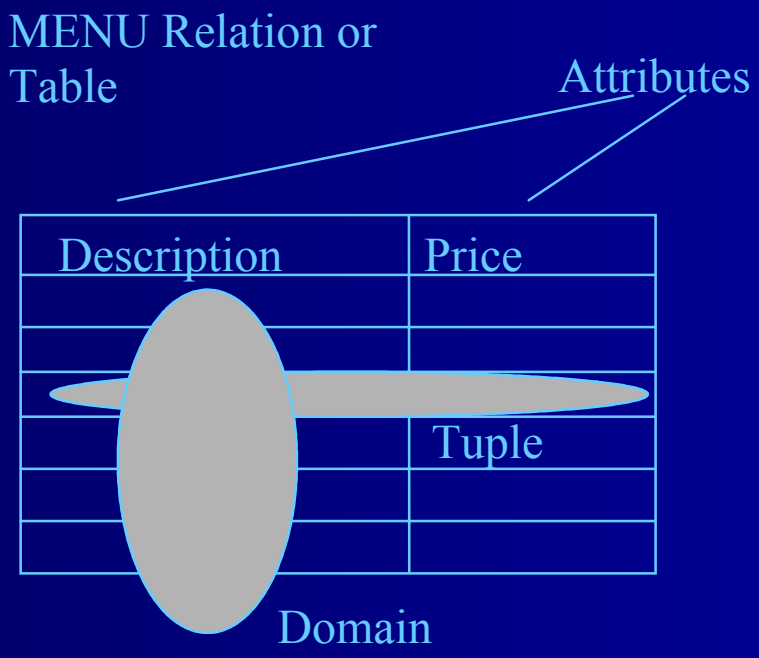
Relational Data Structure



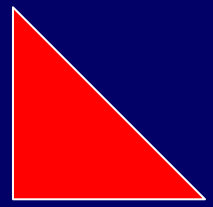
A relational data structure is a collection of tables or relations.

- A relation is a collection of rows or tuples
- A tuple is a collection of columns or attributes
- A domain is a pool of values from which the actual attribute values are taken.

Relational Structure cont

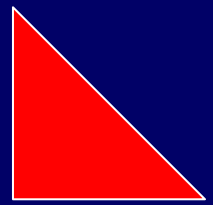


Domain and Integrity Constraints



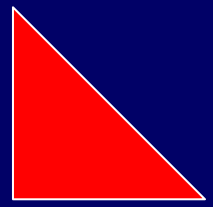
- Domain Constraints
  - limit the range of domain values of an attribute
  - specify uniqueness and 'nullness' of an attribute
  - specify a default value for an attribute when no value is provided.
- Entity Integrity
  - every tuple is uniquely identified by a unique non-null attribute, the primary key.
- Referential Integrity
  - rows in different tables are correctly related by valid key values ('foreign' keys refer to primary keys).

Menu



Description	Price
Large Cola	£0.99
Cheeseburger	£1.99
Burger Royalé	£3.49

External vs Logical

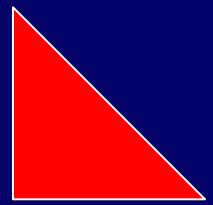


The way a menu is to be shown to a customer may not be the way in which the data is held in a logical model.

The menu model could hold tables about ingredients, individual ingredient costs, overheads, and tax.

The menu provided to a customer is derived from these other tables. It provides a customer-oriented view of the base data.

Columns or Attributes



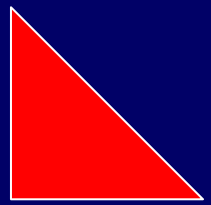
Each column is given a name which is unique within a table

Each column holds data of one specified type. E.g.

integer decimal
character text data
-- the range of values can also be constrained

Some row-column instances may contain no data but instead hold a special null value to indicate that this value is unavailable or inappropriate.

Rows or Tuples



Each row must be uniquely identifiable with a table

- one row records a transaction in the bank case

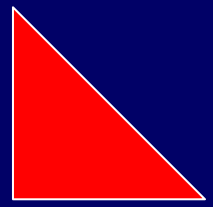
Columns in a specified row may contain no value

- a transaction cannot have credit and debit values simultaneously.

Some columns must contain values for all rows

- date and source, which make the row unique, in the bank account case.

Primary Keys



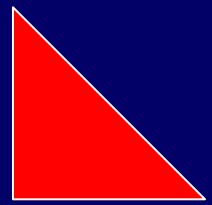
A table requires a key which uniquely identifies each row in the table-entity integrity.

The key could comprise more than one column

A table may have several possible keys, the candidate keys, from which one is chosen as the primary key.

Primary key implies 'UNIQUE NOT NULL'. It may be necessary to generate an artificial primary key if no other unique attribute combination is available within the data.

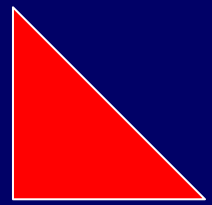
Employee Table - Columns



Full Name	Col name
Employee Number	empno
Forenames	forenames
Address	address
Department No	depno
Surname	surname
Date of Birth	dob
Telephone No	telno

- What is a suitable primary key?
- A suitable key (empno) must be generated since no othe rattributes or combination of attributes uniquely identifies each row.

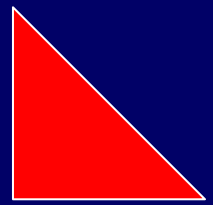
Jobhistory Table - Columns



Full Name	Col name
Employee Number	empno
Start Date	startdate
Salary	salary
Position	position
End Date	enddare

- The primary key combines empno + position to uniquely identify each row.
- empno relates a Jobhistory row to the corresponding Employee row - it is theit is the primary key in the Employee table and a foreign key in the Jobhistory table.

Foreign Keys

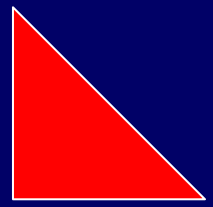


A foreign key is a value held in a table that has exactly the same value as the primary key column of a row in another table.

A foreign key references the primary key of another table and maintains a relationship between the tables.

The column empno (foreign key) in the Jobhistory table must have the same value as one of the empno (primary key) values in the Employee table. This relationship controls referential integrity.

SQL



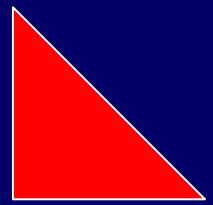
An international Standard Language for manipulating relational databases. It is based on an IBM product called the Structured Query Language.

SQL creates and manipulates tables of data (relations) - it is a data handling language, not a programming language.

A table is a collection of rows (tuples or records).

A row is a collection of columns (attributes).

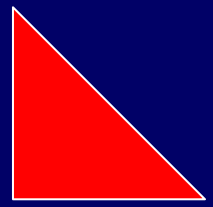
SQL Basics



Basic SQL Statements include:

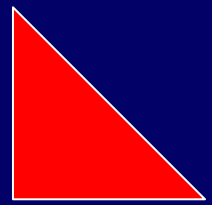
- CREATE - a data structure
- SELECT - read one or more rows from a table
- INSERT - one or more rows into a table
- DELETE - one or more rows from a table
- UPDATE - change the column values in a row
- DROP - a data structure

CREATE table employee



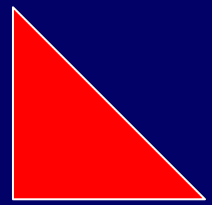
```
CREATE TABLE employee (
    empno       INTEGER         PRIMARY KEY,
    surname     VARCHAR(15),
    forenames   VARCHAR(30),
    dob         date,
    address     VARCHAR(50),
    telno       VARCHAR(50),
    depno       INTEGER REFERENCES department(depno),
        CHECK(dob IS NULL OR
            (dob > '1-jan-1950' AND dob < '31-dec-1980')
        )
);
```

CREATE Table Jobhistory



```
CREATE TABLE jobhistory (
    empno       INTEGER         REFERENCES employee(empno)
    position    VARCHAR(30),
    startdate   date,
    enddate     date,
    salary      DECIMAL(8,2),
            PRIMARY KEY(empno,position)
);
```

SQL SELECT

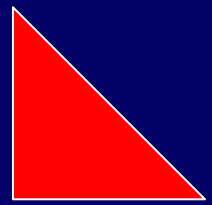


```
SELECT      column-list      -- the simplest SQL SELECT
FROM        table_list;

SELECT      *                -- list ALL employee data
FROM        employee         -- for each employee
;

SELECT      depno,forenames,surname
                             -- list SOME employee
FROM        employee -- data for each employee
;
```

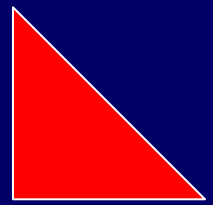
Comparison



```
SELECT          column-list       --
FROM            table_list
[WHERE condition];                -- Comparison Operators:
                                  =,!=,<>,<,<=,>,>=
SELECT          empno,surname
FROM            employee
WHERE           depno = 3;

SELECT          forenames,surname
FROM            employee
WHERE           dob > '2-jan-1958';
```

Comparison cont...

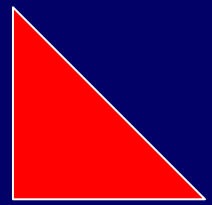


```
SELECT      empno              -- who is or have been
FROM        jobhistory         -- a programmer?
WHERE       position = 'Programmer';

SELECT      empno,position     -- what are employee's
FROM        jobhistory         -- current positions?
WHERE       enddate IS NULL;
```

Note that NULL indicates a value which is missing, not known, inappropriate, etc. NULL is not a blank or zero. NULL cannot be tested for equality with other NULL values.

SELECT with BETWEEN

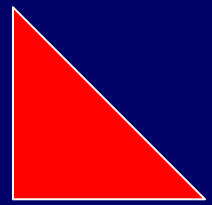


```
SELECT      empno,surname,forenames,dob
FROM        employee
WHERE       dob BETWEEN '30-jun-1954' AND '1-jan-1959';
```

Note that the BETWEEN predicate is inclusive. The above condition is equivalent to :

```
WHERE       dob >= '30-jun-1954' AND <='1-jan-1959';
```

Pattern Matching

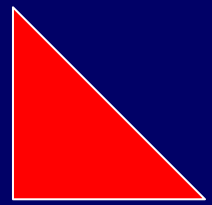


Simple pattern matching is carried out using LIKE:
LIKE 'pattern-to-match'
Where the pattern can include special wildcard characters:

% 0 or more arbitrary characters
_ any one character

```
SELECT      forenames, surname, address
FROM        employee
WHERE       address LIKE '%Edinburgh%';
```

ORDER and DISTINCT



```
SELECT [DISTINCT]    column_list
FROM                 table_list
[WHERE               condition]
[ORDER BY            attribute[ DESC/ASC]
                     [,attribute [DESC,ASC]]...];
```

Note that ASCending order is the default

```
SELECT DISTINCT      empno
FROM                 jobhistory
WHERE                starddate < '1-jan-1980'
ORDER BY             empno DESC;
```