COMP810 Data Warehousing and Big Data

Semester 2 2024

Dr Victor Miranda

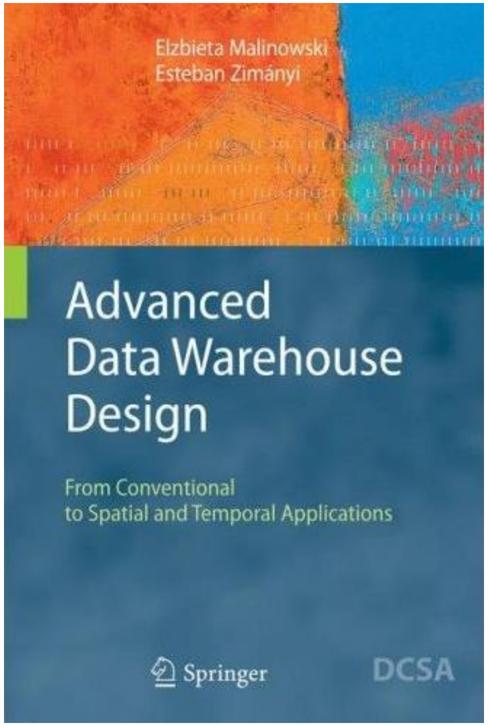
COMP810

Week 1 Data Warehousing

- Database Concepts
- Data Warehouse Concepts
- Introduction to SQL

Chapter 2 of Book

Databases & Data Warehouses



Elzbieta Malinowski
Esteban Zimányi

Advanced
Data Warehouse
Design

From Conventional
to Spatial and Temporal Applications

Springer

DCSA

Outline (Lecture 65 min + Lab 45 min)

- Motivation
- Database Concepts
  - Concepts of Data and Database
  - Database Management System
- Introduction to SQL
  - SELECT
  - CREATE TABLE
  - WHERE
  - Operators
  - Domain types

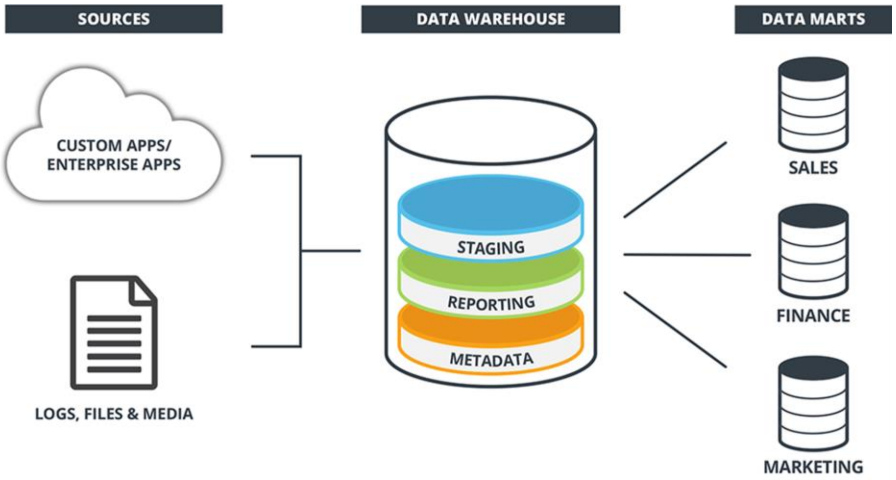
Why Are We in this Course?

In a nutshell, a data warehouse is a management system that is designed to enable and support business intelligence (BI) activities, especially analytics

Why Are We in this Course?

In a nutshell, a data warehouse is a management system that is designed to enable and support business intelligence (BI) activities, especially analytics

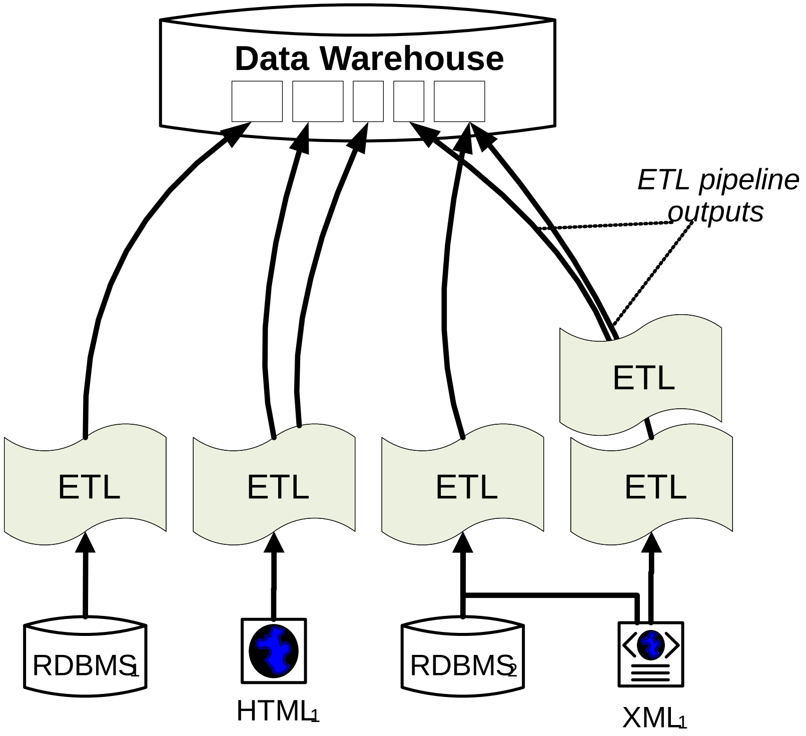
> Data warehouse platforms also sort data based on different subject matter, such as customers, products or business activities.



- Why is data warehousing important?
  - Ensure consistency
  - Make better business decisions
  - Quickly access

Why Are We in this Course?

- At the top – a centralized database
  - Generally configured for queries and appends – not transactions
  - Many indices, materialized views, etc.
- Data is loaded and periodically updated via **Extract/Transform/Load (ETL) tools**



Concepts of Data & Database

- Data: Any numerical, character or other symbols that can be ‘recorded’ for further processing, usually, a computer.

Essentially, information (representation) of facts and numbers used to analyse something or make decisions.

> Data type

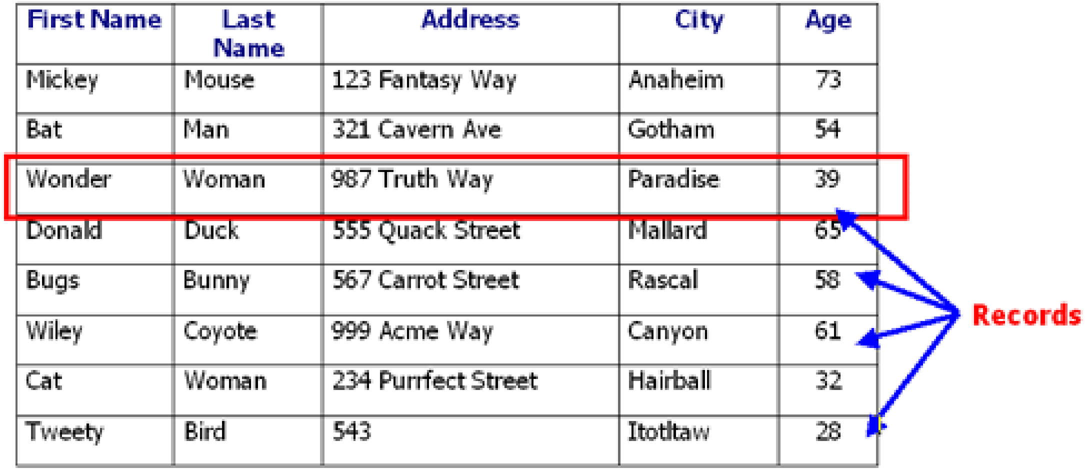
> Format

> Storage requirements

Concepts of Data & Database

- Data: Any numerical, character or other symbols that can be 'recorded' for further processing, usually, a computer.

- Record: Simply a set of data stored in a table, for example a *customer record*. It can contain one or more 'values'.



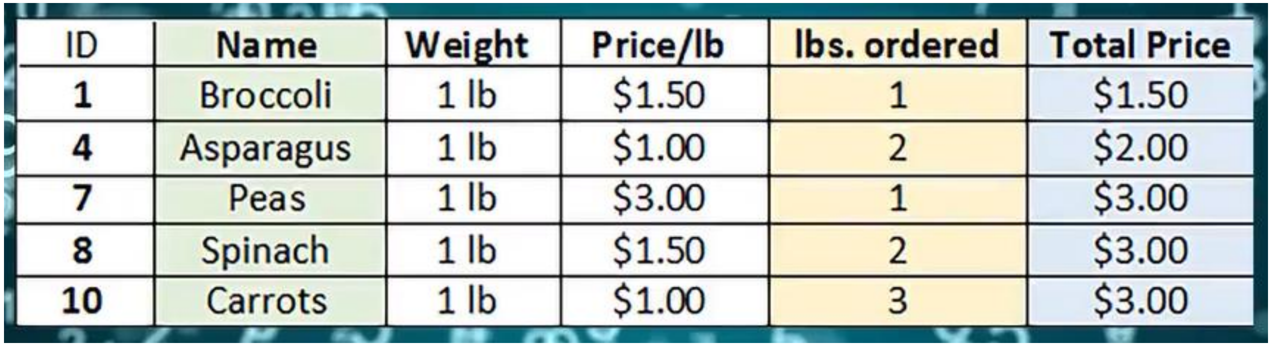
First Name	Last Name	Address	City	Age
Mickey	Mouse	123 Fantasy Way	Anaheim	73
Bat	Man	321 Cavern Ave	Gotham	54
Wonder	Woman	987 Truth Way	Paradise	39
Donald	Duck	555 Quack Street	Mallard	65
Bugs	Bunny	567 Carrot Street	Rascal	58
Wiley	Coyote	999 Acme Way	Canyon	61
Cat	Woman	234 Purrfect Street	Hairball	32
Tweety	Bird	543	Itotltaw	28

Records

Concepts of Data & Database

- Table: It's an 'object' (an arrangement) containing groups of records. The table defines the data that each record may contain, according to each 'field'.

We usually think of a 'table' like a rectangular arrangement of records (in rows) and their attributes (columns) .

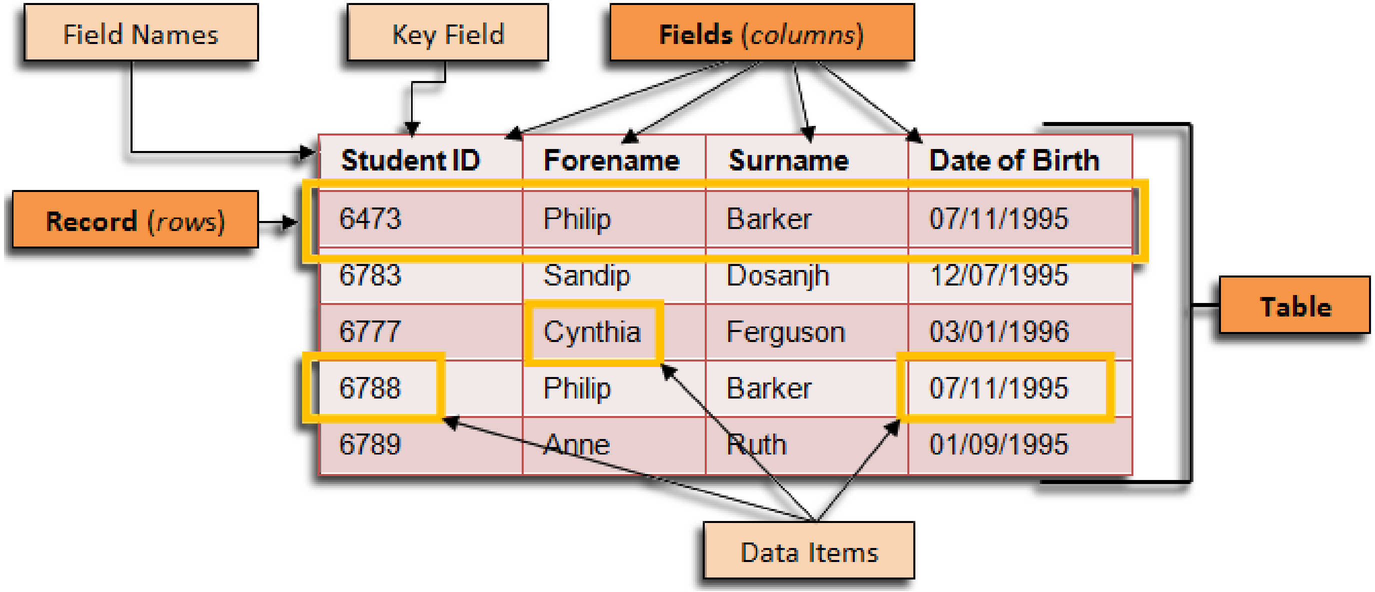


ID	Name	Weight	Price/lb	lbs. ordered	Total Price
1	Broccoli	1 lb	$1.50	1	$1.50
4	Asparagus	1 lb	$1.00	2	$2.00
7	Peas	1 lb	$3.00	1	$3.00
8	Spinach	1 lb	$1.50	2	$3.00
10	Carrots	1 lb	$1.00	3	$3.00

Concepts of Data & Database

- Table: It's an 'object' (an arrangement) containing groups of records. The table defines the data that each record may contain, according to each 'field'.

- Field: An 'attribute' of a record in a table (a column).



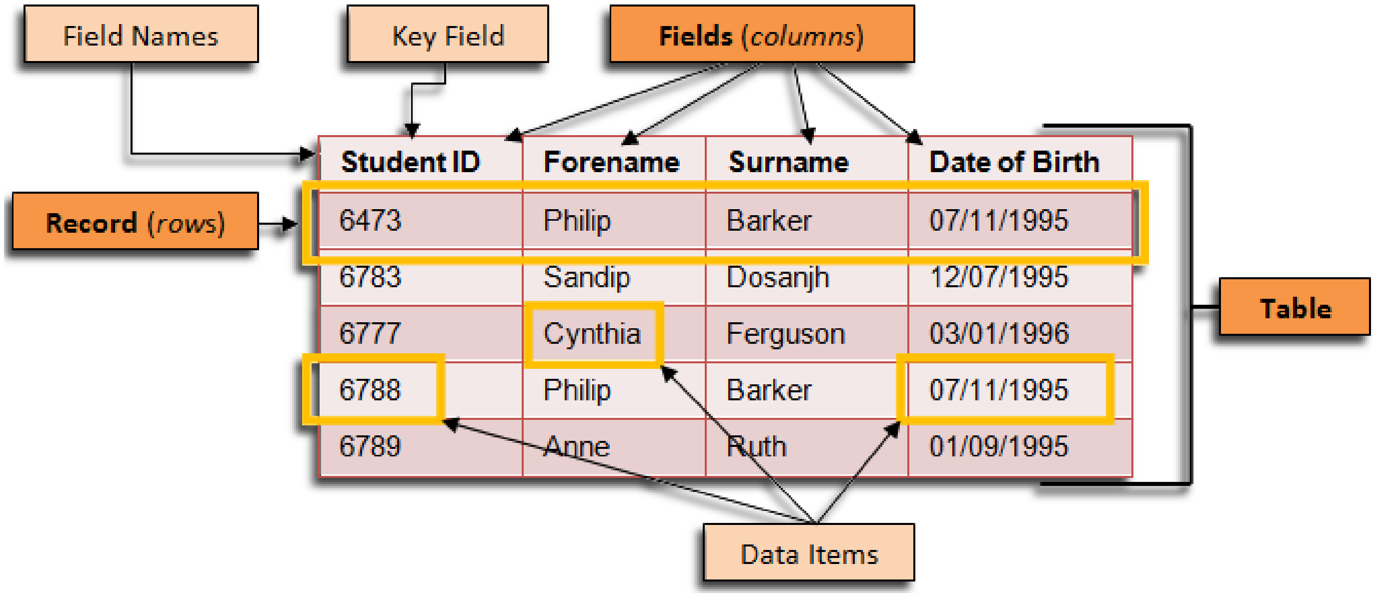
Student ID	Forename	Surname	Date of Birth
6473	Philip	Barker	07/11/1995
6783	Sandip	Dosanjh	12/07/1995
6777	Cynthia	Ferguson	03/01/1996
6788	Philip	Barker	07/11/1995
6789	Anne	Ruth	01/09/1995

Concepts of Data & Database

Think of the rows and columns of a typical spreadsheet.

- Rows are horizontal and go across the spreadsheet from left to right. These are our 'records'. Every new row creates a new row-entry (record)

- Columns, on the other hand, are vertical and flow down the spreadsheet. These are our 'fields'.

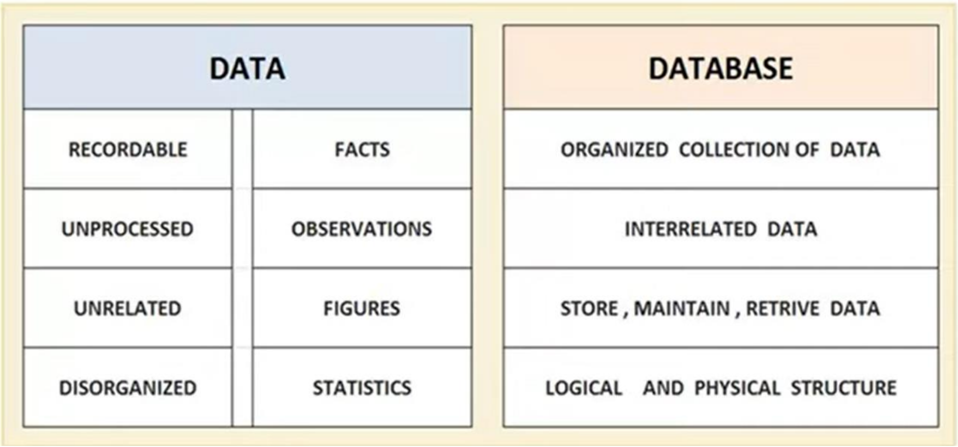


Student ID	Forename	Surname	Date of Birth
6473	Philip	Barker	07/11/1995
6783	Sandip	Dosanjh	12/07/1995
6777	Cynthia	Ferguson	03/01/1996
6788	Philip	Barker	07/11/1995
6789	Anne	Ruth	01/09/1995

Concepts of Data & Database

- Database: A collection of multiple, inter-related, TABLES specially organized for rapid search and retrieval by a computer.

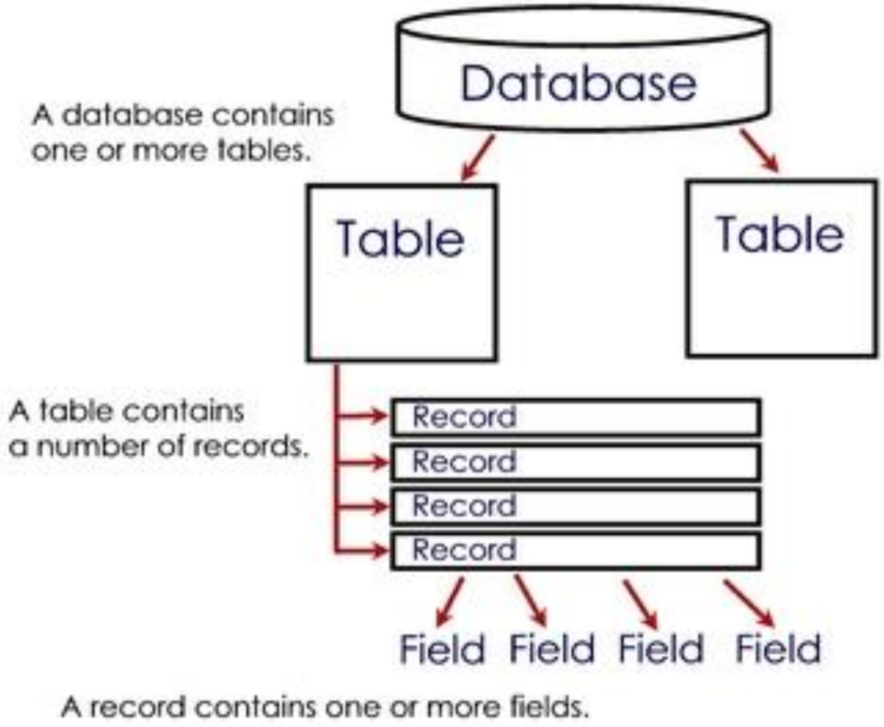
For a given database, there are multiple tables, each containing multiple records.



DATA		DATABASE
RECORDABLE	FACTS	ORGANIZED COLLECTION OF DATA
UNPROCESSED	OBSERVATIONS	INTERRELATED DATA
UNRELATED	FIGURES	STORE , MAINTAIN , RETRIVE DATA
DISORGANIZED	STATISTICS	LOGICAL AND PHYSICAL STRUCTURE

Concepts of Data & Database

SUMMARY: Records are stored in rows that make up the table which in turn make up the database.



Database
A database contains one or more tables.
Table
Table
A table contains a number of records.
Record
Record
Record
Record
Field
Field
Field
Field
A record contains one or more fields.

Concepts of Data & Database

NOTE:

(a) A fancy word for a database record is a ‘ tuple ’.

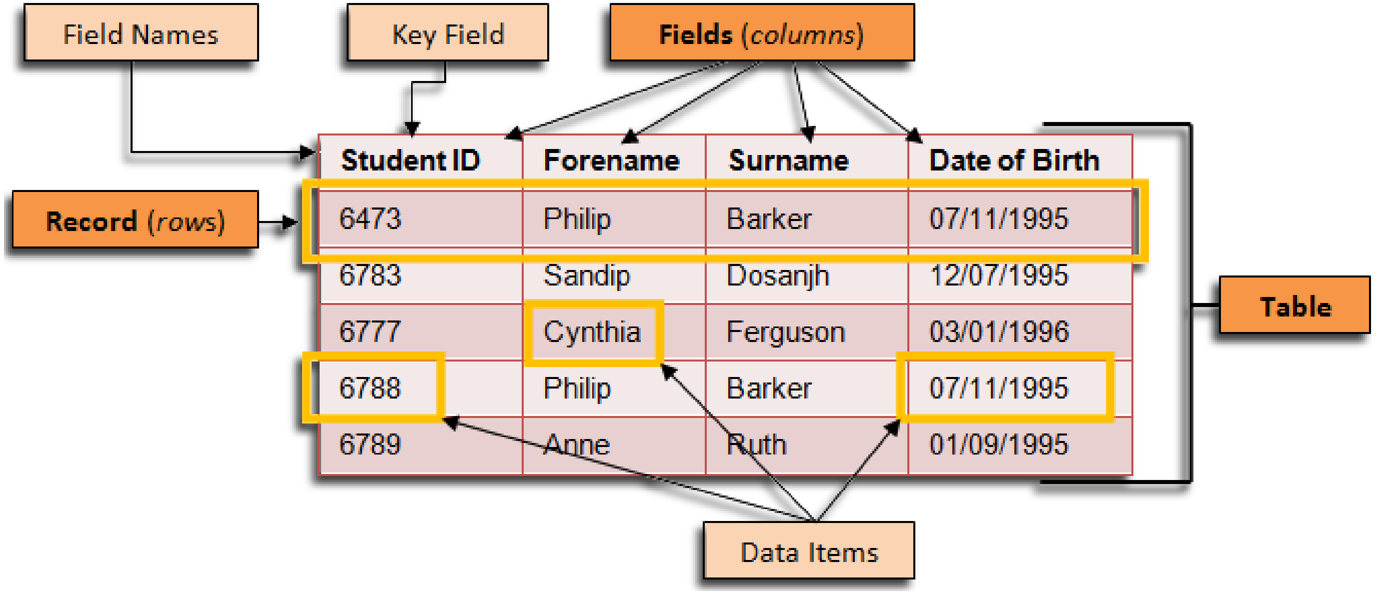
In this course we’ll use this term to refer to a ‘record’.

(b) A column can be called an ‘attribute’.

Concepts of Data & Database

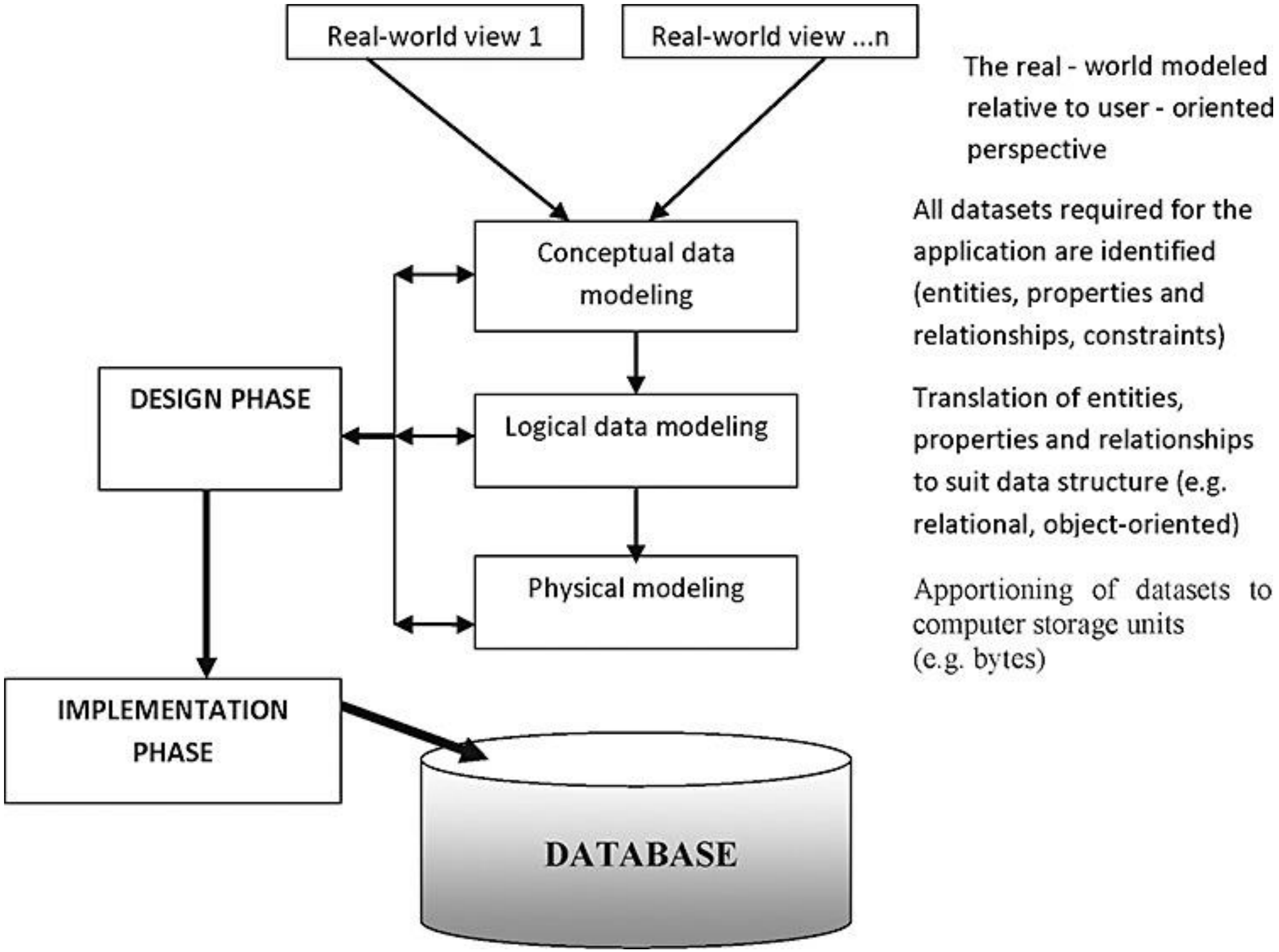
- Records provide a practical way to store and retrieve data from the database.

- Each record can have different kinds of data; thus a single row could have different types of information.



Student ID	Forename	Surname	Date of Birth
6473	Philip	Barker	07/11/1995
6783	Sandip	Dosanjh	12/07/1995
6777	Cynthia	Ferguson	03/01/1996
6788	Philip	Barker	07/11/1995
6789	Anne	Ruth	01/09/1995

Steps in Database Design



Real-world view 1

Real-world view ...n

The real - world modeled relative to user - oriented perspective

Conceptual data modeling

All datasets required for the application are identified (entities, properties and relationships, constraints)

DESIGN PHASE

Logical data modeling

Translation of entities, properties and relationships to suit data structure (e.g. relational, object-oriented)

Physical modeling

Apportioning of datasets to computer storage units (e.g. bytes)

IMPLEMENTATION PHASE

DATABASE

Steps in Database Design

- Requirements specification: Collects information about users' needs with respect to the database system
- Conceptual design: Builds a user-oriented representation of the database without any implementation considerations
- Logical design: Translates the conceptual schema from the previous phase into an implementation model common to several DBMSs, e.g., relational or object-relational
- Physical design: Customizes the logical schema from the previous phase to a particular platform, e.g., Oracle or SQL Server

Types of databases

a. Relational Database ⇦

b. Object – relational (oriented) database ⇦

c. Hierarchical Databases

d. Non-relational databases

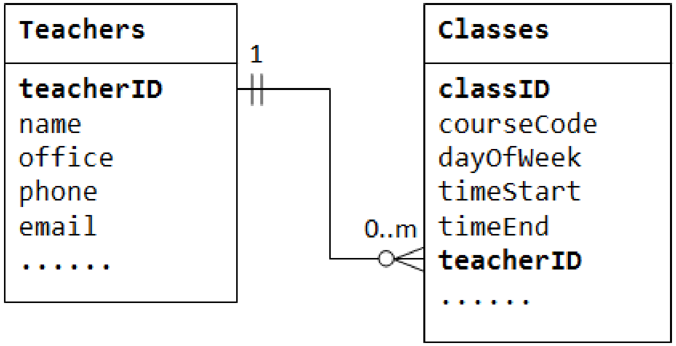
Relational Database

‘DEFINITION’

A relational database stores various types of information related to each other, typically organized in tables consisting of rows and columns. Tables are manipulated with queries in a query language.

For example, a typical business order entry database would include a table that describes a customer with columns for name, address,

Toy examples:

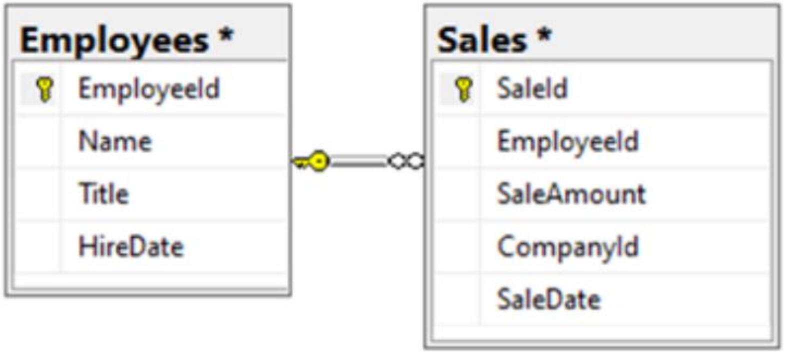


Teachers

teacherID
name
office
phone
email
......

Classes

classID
courseCode
dayOfWeek
timeStart
timeEnd
teacherID
......



Employees *

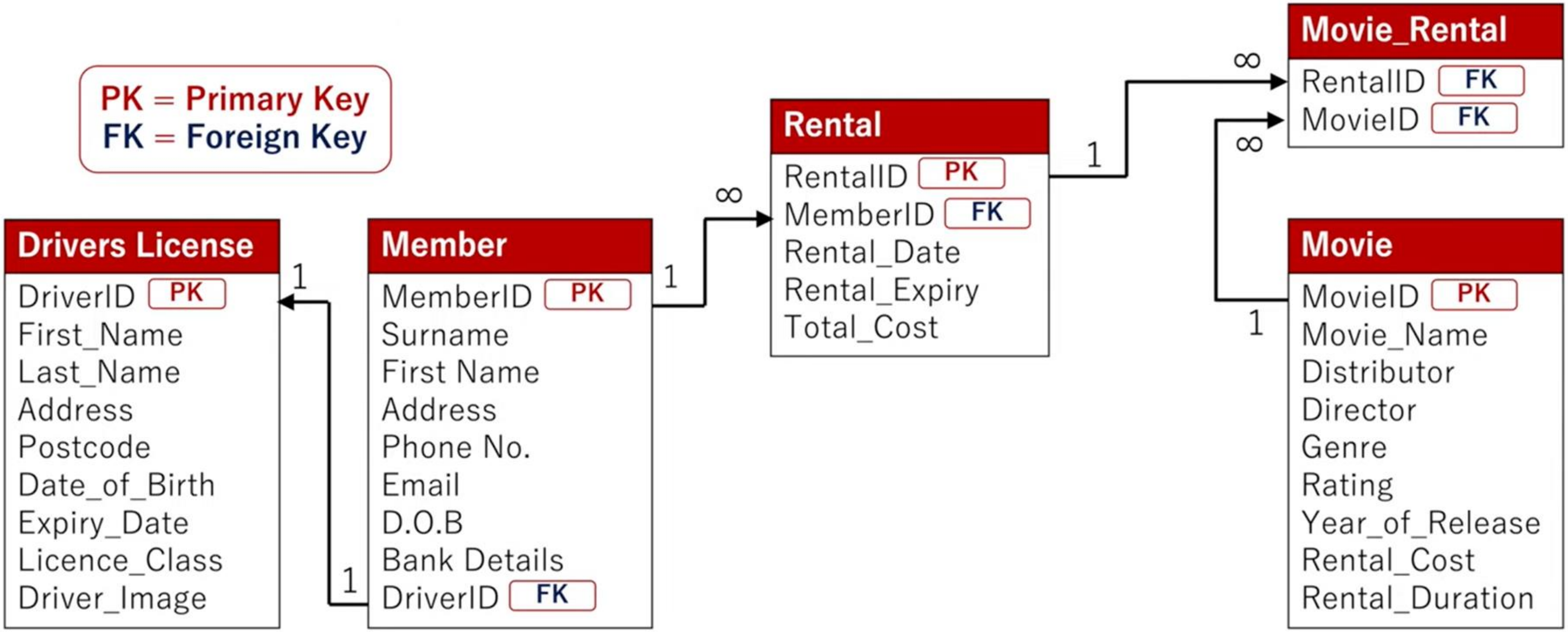
EmployeeId
Name
Title
HireDate

Sales *

SaleId
EmployeeId
SaleAmount
CompanyId
SaleDate

Relational Database: Movie Rental System Database Ex.

Example: Three types of relationships



PK = Primary Key
FK = Foreign Key

Drivers License
DriverID PK
First_Name
Last_Name
Address
Postcode
Date_of_Birth
Expiry_Date
Licence_Class
Driver_Image

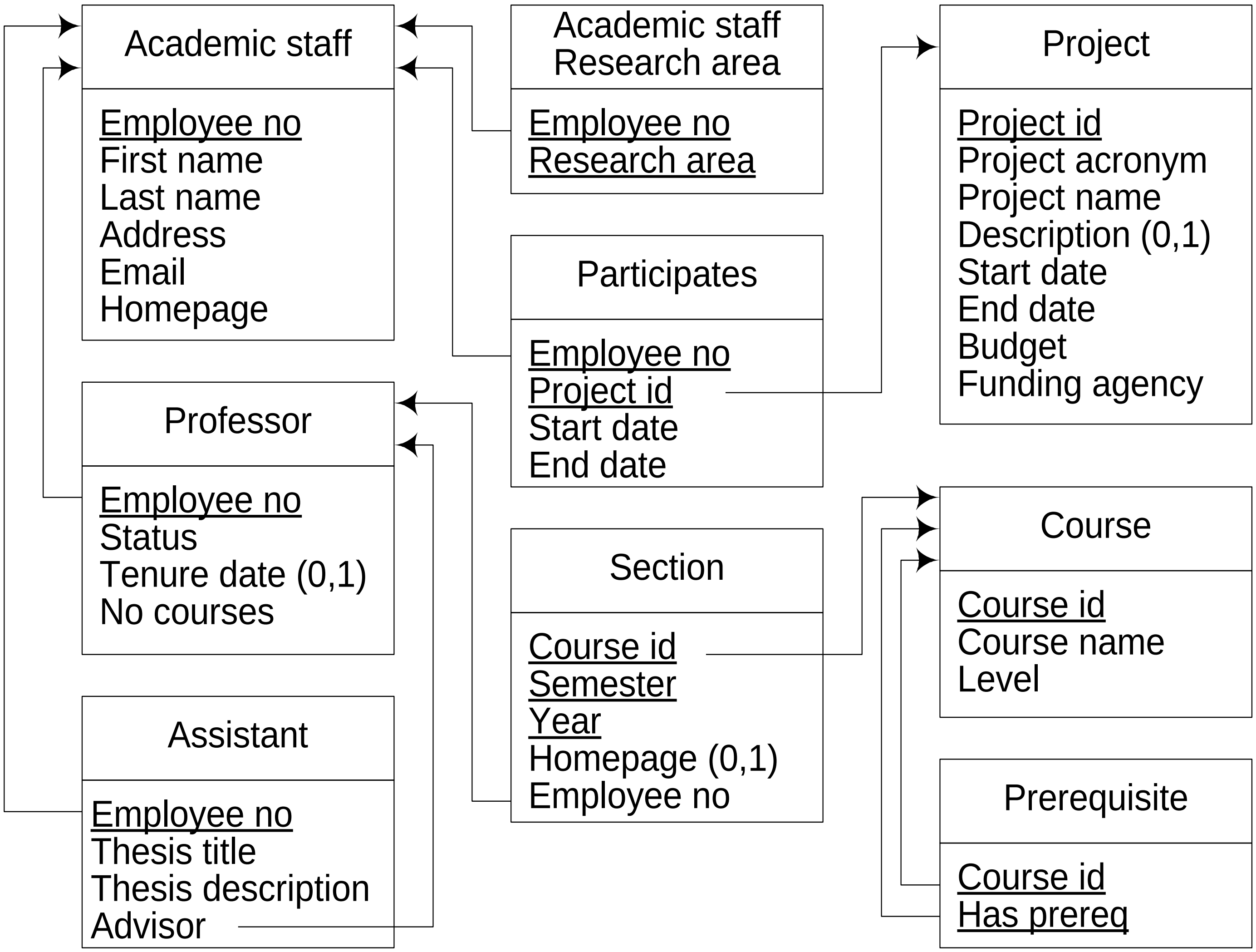
Member
MemberID PK
Surname
First Name
Address
Phone No.
Email
D.O.B
Bank Details
DriverID FK

Rental
RentalID PK
MemberID FK
Rental_Date
Rental_Expiry
Total_Cost

Movie_Rental
RentalID FK
MovieID FK

Movie
MovieID PK
Movie_Name
Distributor
Director
Genre
Rating
Year_of_Release
Rental_Cost
Rental_Duration

Relational Database: University Example



Academic staff
Employee no
First name
Last name
Address
Email
Homepage

Academic staff Research area
Employee no
Research area

Project
Project id
Project acronym
Project name
Description (0,1)
Start date
End date
Budget
Funding agency

Participates
Employee no
Project id
Start date
End date

Professor
Employee no
Status
Tenure date (0,1)
No courses

Section
Course id
Semester
Year
Homepage (0,1)
Employee no

Course
Course id
Course name
Level

Assistant
Employee no
Thesis title
Thesis description
Advisor

Prerequisite
Course id
Has prereq

Object – relational Database (ORDB)

‘DEFINITION’

An object – relational database (ORDBs) are a hybrid between traditional relational database but with an object-oriented database model.

The data resides in the database and is manipulated collectively with queries in a query language. This model is used to represent real-world entities. The data and data relationship are stored together in a single entity known as an object in the Object Oriented Model.

Object – relational Database (ORDB)

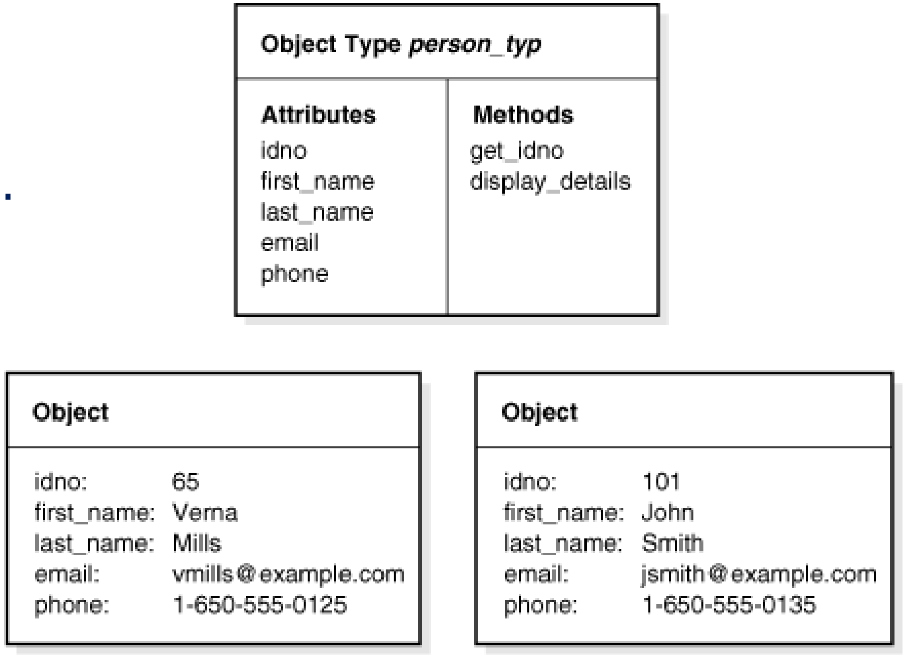
‘DEFINITION’

An object – relational database (ORDBs) are a hybrid between traditional relational database but with an object-oriented database model.

Example 1:

Each ‘row’ is an object (person_typ).

We have two ‘rows’ (each one is a table in the usual way) of objects.



Object Type *person_typ*

Attributes
idno
first_name
last_name
email
phone

Methods
get_idno
display_details

Object

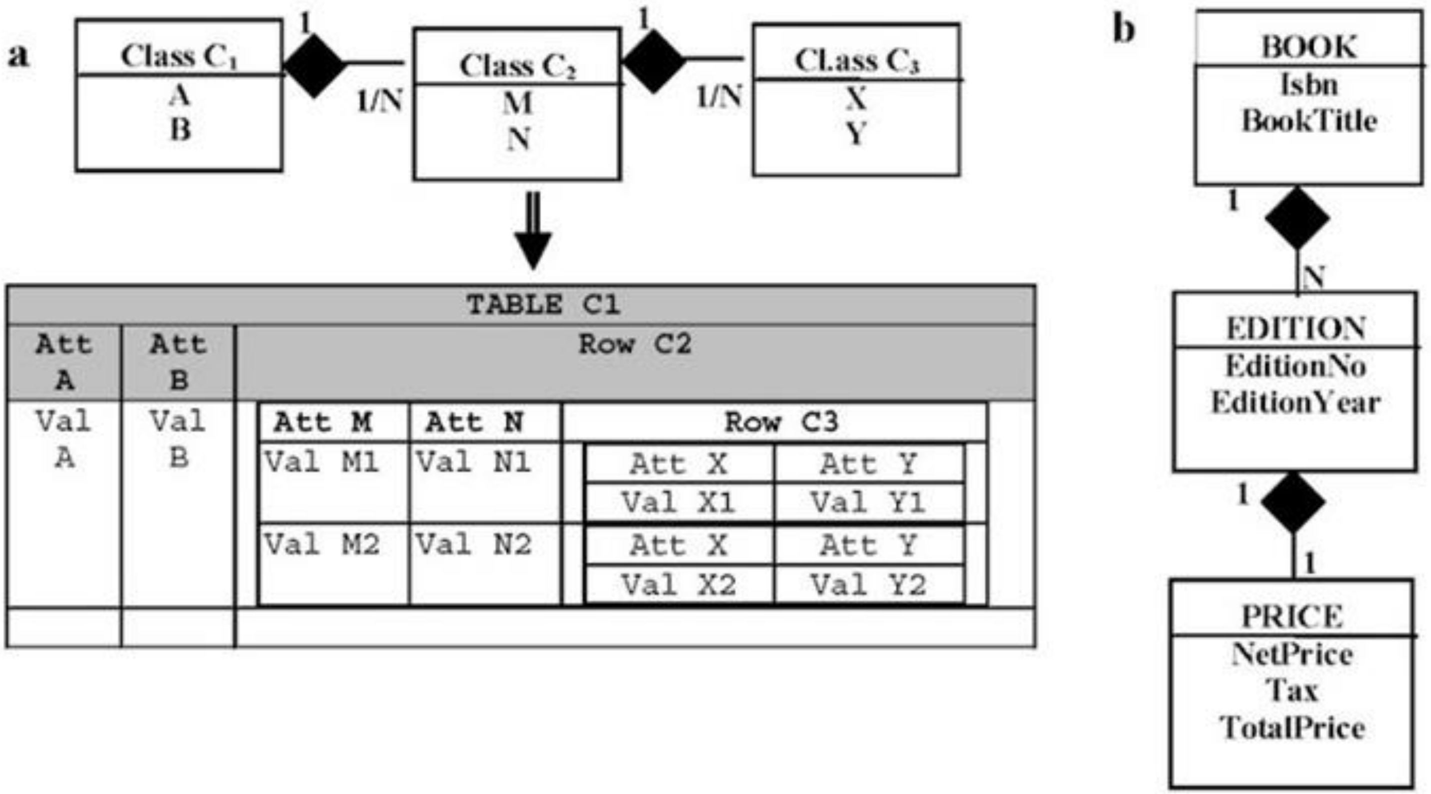
idno:	65
first_name:	Verna
last_name:	Mills
email:	vmills@example.com
phone:	1-650-555-0125

Object

idno:	101
first_name:	John
last_name:	Smith
email:	jsmith@example.com
phone:	1-650-555-0135

Object – relational Database (ORDB)

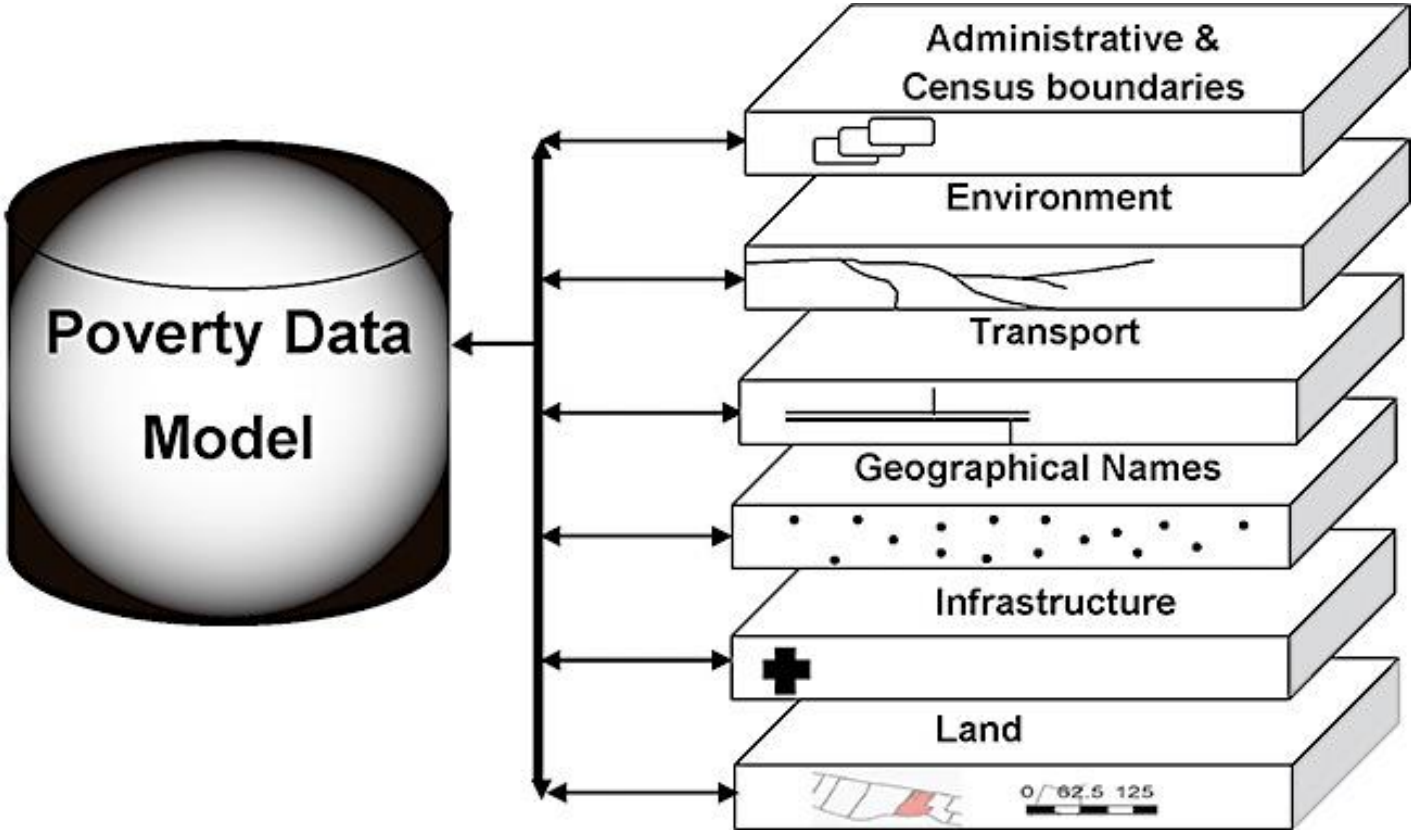
Example 2:



<table>
<tr><th colspan="3">TABLE C1</th></tr>
<tr><th>Att A</th><th>Att B</th><th>Row C2</th></tr>
<tr><td>Val A</td><td>Val B</td><td>
<table>
<tr><th>Att M</th><th>Att N</th><th>Row C3</th></tr>
<tr><td>Val M1</td><td>Val N1</td><td><table><tr><th>Att X</th><th>Att Y</th></tr><tr><td>Val X1</td><td>Val Y1</td></tr></table></td></tr>
<tr><td>Val M2</td><td>Val N2</td><td><table><tr><th>Att X</th><th>Att Y</th></tr><tr><td>Val X2</td><td>Val Y2</td></tr></table></td></tr>
</table>
</td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

Object – relational Database (ORDB)

Example 3:



Reference (a)

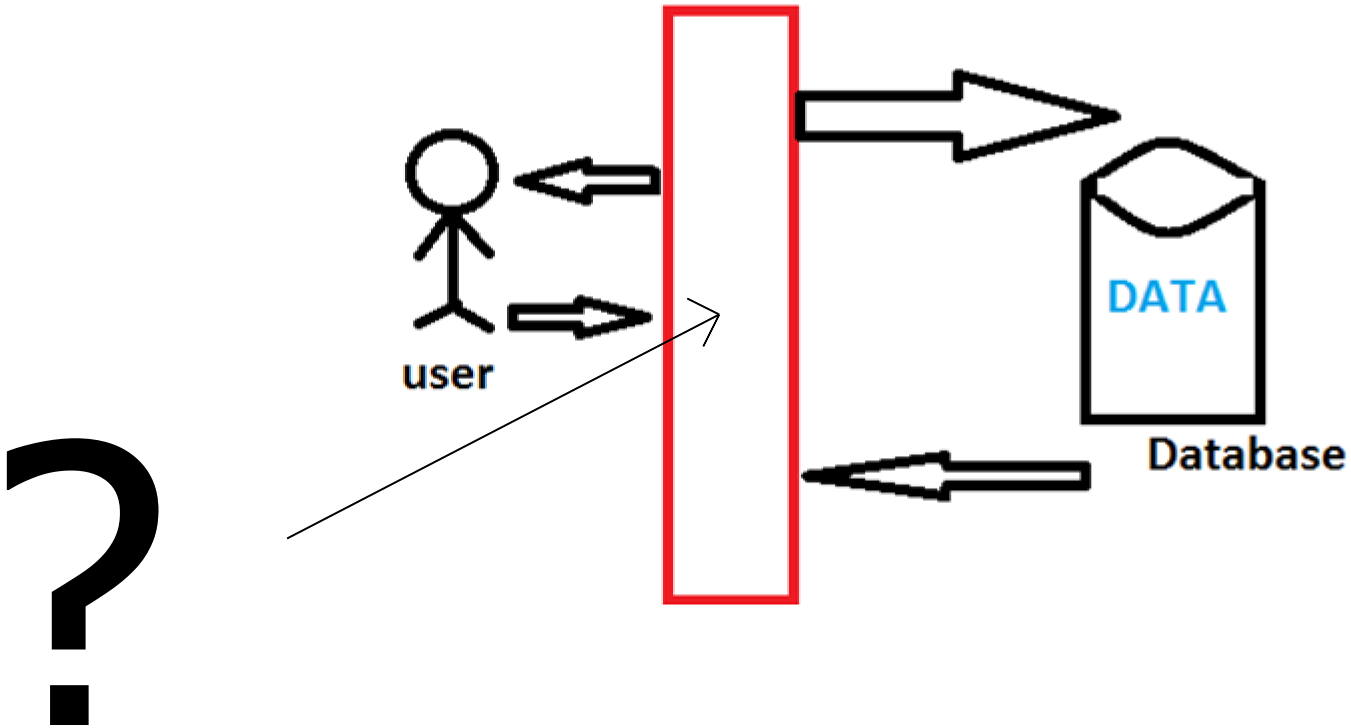
Two questions

Q1 :

?

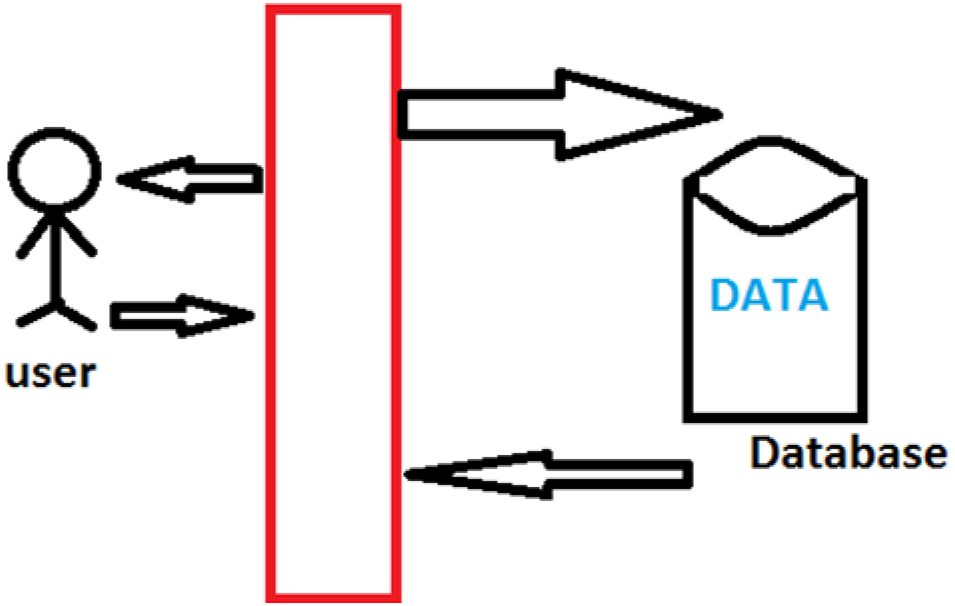
Two questions

Q2 :



Two questions

Q2 :



We need:

(a) Q1 - database management system (DBMS), and

(b) Q2 - an interface

Database Management System

DEFINITION

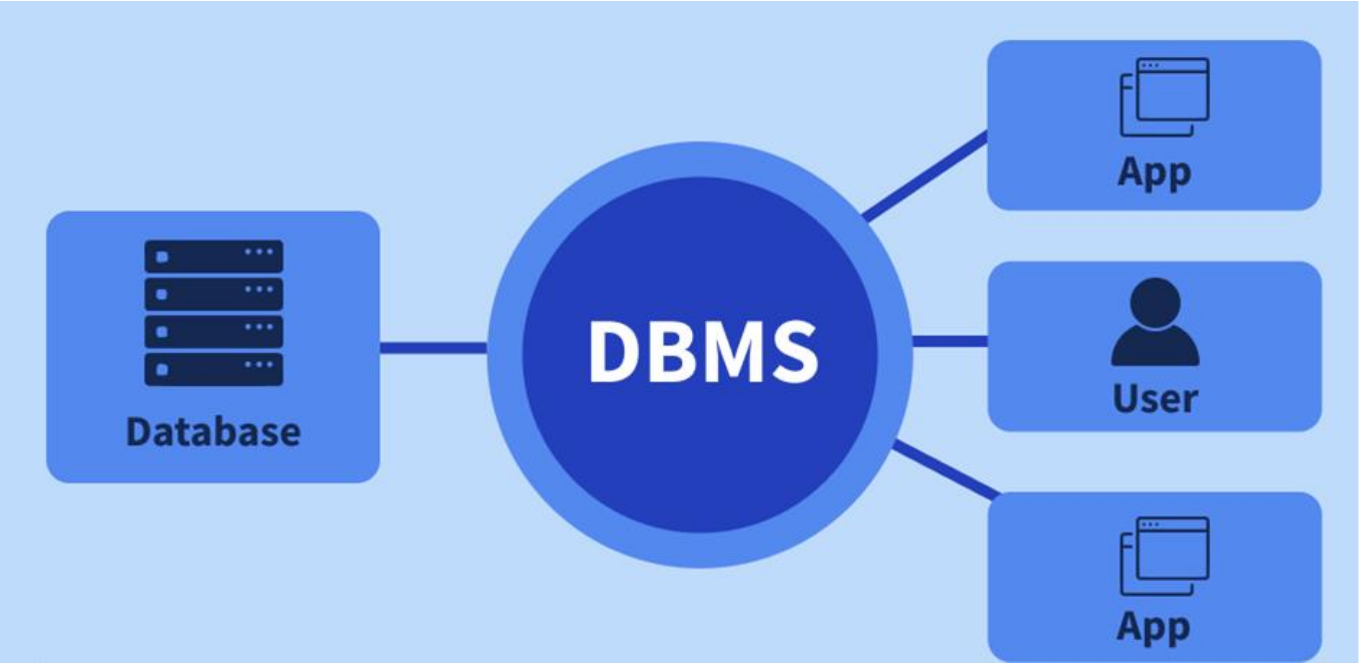
A database management system (DBMS) is a software based-system that enables to:

- specify the structure of a database,
- create, query and modify the data in the database, and
- control access to the database, (and)
- analyse the data (in some cases).

It's like the modern version of keeping files filled with vital information.

Database Management System

In short: Database Management System (DBMS) – software system that supports creation, population, and querying of a database



Database Management System

ADVANTAGES OF DBMSs

A database management system (DBMS) is a software based-system that enables to:

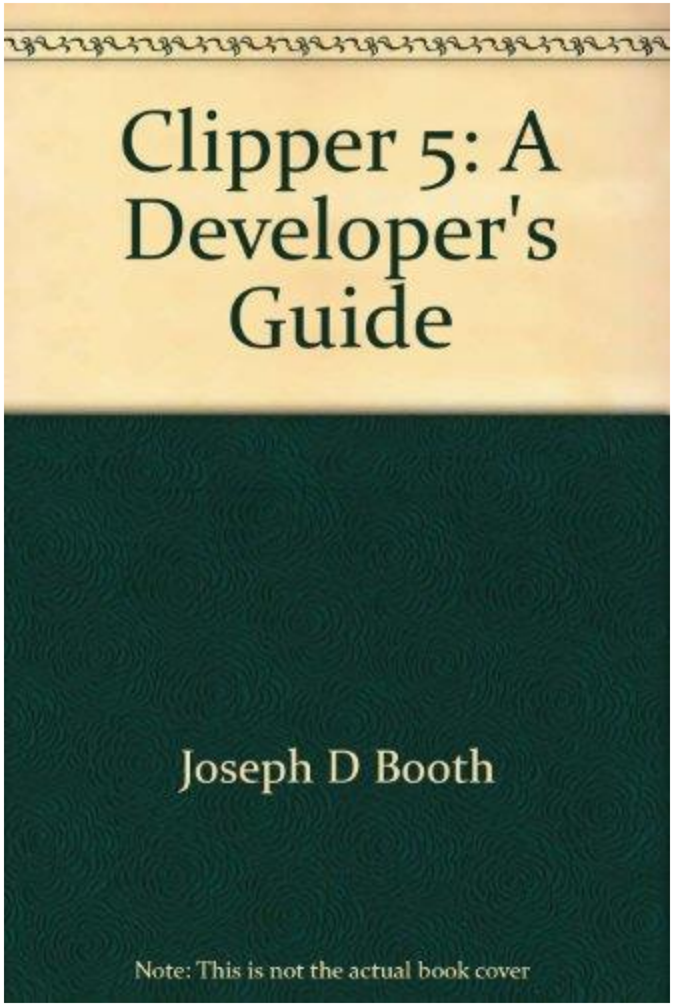
- It helps maintain data uniformity
- Handles large set of data efficiently
- Versatile
- Faster way of managing data.

Examples: FoxPro, Clipper DBMS, RDBMS, etc.

About Clipper DBMS

Clipper (1997) is a database application that uses a dBASE format, operated primarily under MS-DOS. The Clipper connector primary data file usually has a .DBF extension and the memo file has a .DBT extension.

>> Clipper files are structured, i.e., both the data and the file structure are stored inside the primary data file.


Clipper 5: A Developer's Guide
Joseph D Booth
Note: This is not the actual book cover

Database Management System

ADVANTAGES OF DBMSs

A database management system (DBMS) is a software based-system that enables to:

- It helps maintain data uniformity
- Handles large set of data efficiently
- Versatile
- Faster way of managing data.

Examples: FoxPro, Clipper DBMS, RDBMS, etc.

Relational Database Management System

DEFINITION

Relational Database Management Systems (RDBMS) are simply advanced versions of DBMS specially design for creating and managing relational databases (relational models).

In short, it is connection-software that operates on a relational schema (database arranged in tables with rows and columns).

Relational Database Management System

An RDBMS offers businesses a systematic view of data, which can be used to enhance different aspects of decision-making. Relational databases offer a number of other advantages as well, including:

- Allow multiple-user access
- Store large packs of data
- Maintains Data Integration
- Better Tools for Structuring and Organizing Data

Let’s answer the questions:

Q2 :

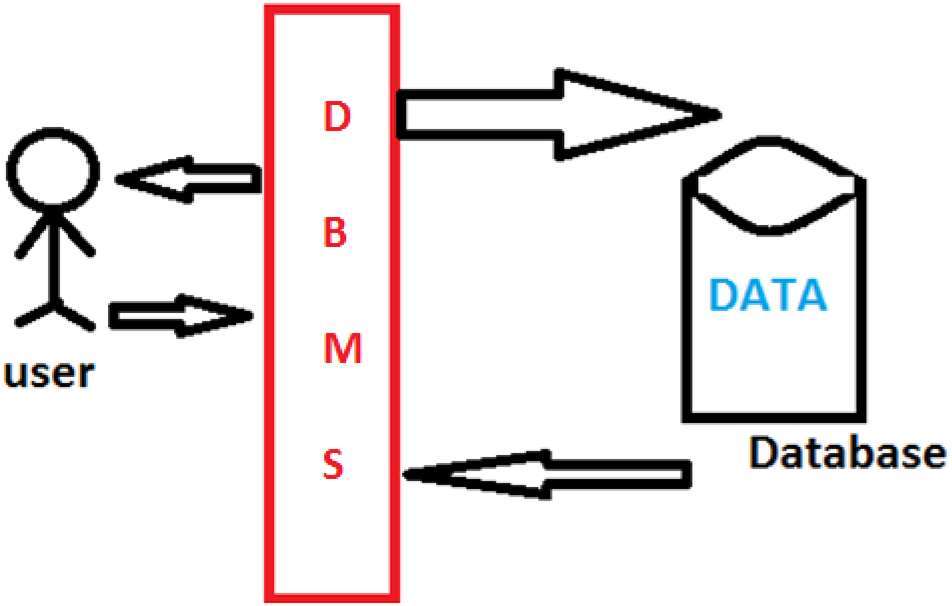
We need:

(a) Q1 - database management system (DBMS), and

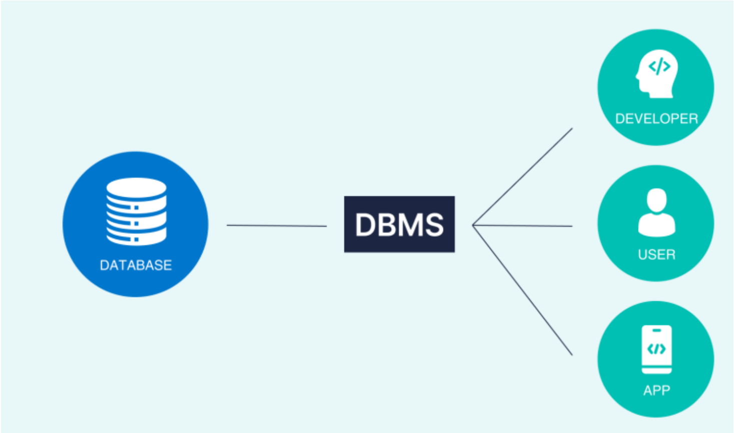
(b) Q2 - an interface

ANSWER: We needs a DBMS & an interface

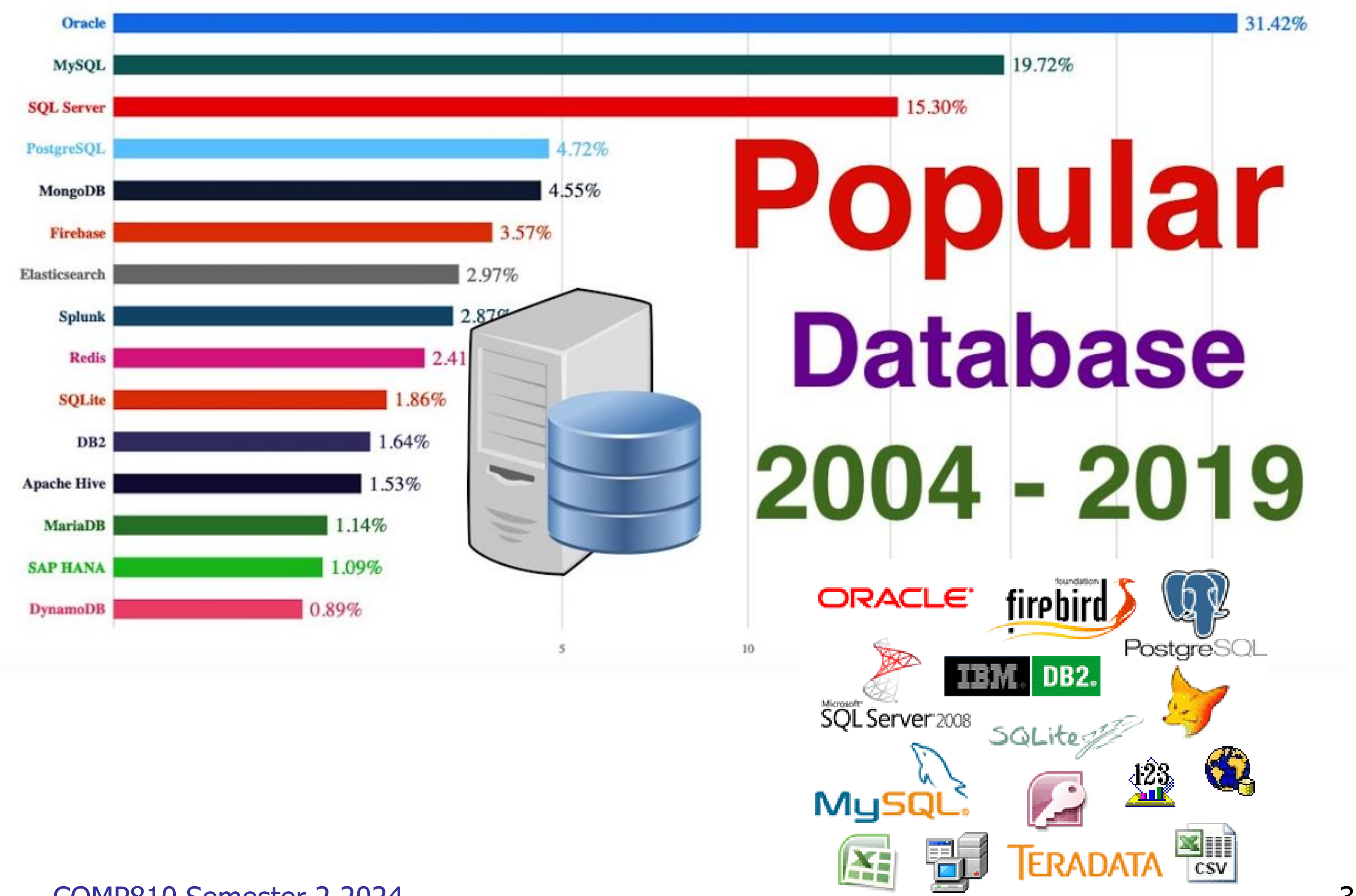
User communicates to the DBMS interface. DBMS fulfils requirements



DBMS = Different software to insert, delete, update, query database

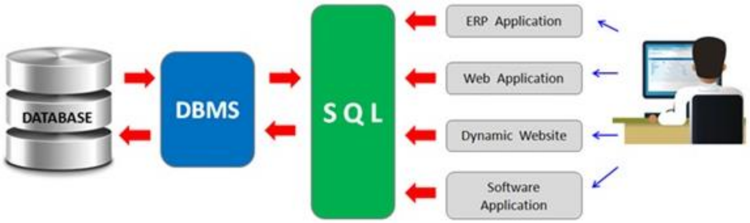


Most popular RDBMS 2004 - 2019



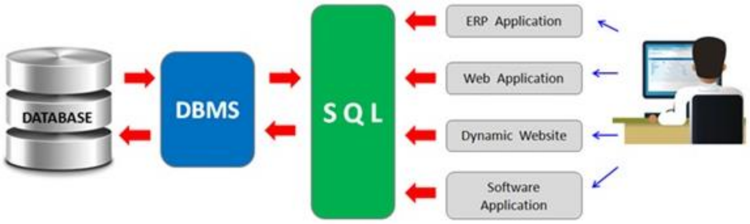
In this course, we will adopt SQL

Why SQL?



- SQL is a high-level query language.
  - Expresses "what to do" rather than "how to do it.". 'Free' format
  - Avoid a lot of data-manipulation details needed in procedural languages like C++ or Java.
  - Original name 'SeQueL'
- Database management system figures out "best" way to execute query.
  - Called *query optimization*

In this course, we will adopt SQL



- SQL for getting information from a database:
  - Data Definition Language (DDL) – We will create /alter / delete tables and their attributes
  - Data Manipulation Language (DML) – Insertion, Modification, Deletion of tuples in tables
  - Data retrieval – Perform simple and complex queries

- Many standards out there:

ANSI SQL, SQL92 (a.k.a. SQL2), SQL99 (a.k.a. SQL3), ….

Table name

Attribute names

Tables in SQL

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Tuples or rows

‘Schema’ of an SQL Table & Key attributes

- The *schema* of a table is the table name and its attributes – Notation:

  Product(PName, Price, Category, Manfacturer)

- A *key* is an attribute whose values are unique; we underline a key – Notation:

Product(PName, Price, Category, Manfacturer)

DDL

The SQL data-definition language (DDL) allows the specification of information about relations (tables), including:

- The schema for each relation.
- The domain of values associated with each attribute.
- Integrity constraints

DOMAIN Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n.*
- **varchar(n).** Variable length character strings, with user-specified maximum length *n.*
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point. (ex., **numeric**(3,1), allows 44.5 to be stores exactly, but not 444.5 or 0.32)
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.

CREATE TABLE command

- An SQL relation is defined using the **create table** command:

  create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

  - *r* is the **name** of the relation (character string)
  - each A_i is an **attribute** name in the schema of relation *r*
  - D_i is the data type of values in the domain of attribute A_i

- Example:

  create table *instructor* (
  ID **char**(5),
  name **varchar**(20)**,**
  dept_name **varchar**(20),
  salary **numeric**(8,2))

Integrity Constraints: Primary Key vs Foreign Key

- An SQL relation is defined using the **create table** command:

```
create table instructor (
      ID            char(5),
      name          varchar(20),
      dept_name     varchar(20),
      salary        numeric(8,2),

      primary key ( ZZZZ, ZZZZ, ZZZ ),
      foreign key (ZZZZ) references r)
```

- A primary key uniquely identifies a row in a table. Cannot have a NULL value

- A foreign key is used to link two tables together by referencing the primary key of the related table.

CREATE TABLE – Integrity Constraints

FORMAT:

- **primary key** (A_1, ..., A_n)
- **foreign key** (A_m, ..., A_n) **references** *r*

Example:

```
create table instructor (
        ID             char(5),
        name           varchar(20) not null,
        dept_name      varchar(20),
        salary         numeric(8,2),
        primary key (ID),
        foreign key (dept_name) references department);
```

primary key declaration on an attribute automatically ensures **not null**

Basic Query Structure – SELECT statement

```
SELECT  <attributes>
FROM     <one or more relations>
WHERE  <conditions>
```

Basic Query Structure – SELECT statement

- A typical SQL query has the form:

  select $A_1, A_2, \ldots, A_n$
  from $r_1, r_2, \ldots, r_m$
  where P

  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate.

```
SELECT  <attributes>
FROM    <one or more relations>
WHERE   <conditions>
```

Basic Query Structure – SELECT statement

- A typical SQL query has the form:

  select $A_1, A_2, ..., A_n$
  from $r_1, r_2, ..., r_m$
  where P

  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate.

- The result of an SQL query is a relation.

SELECT clause (in practice)

“selection”

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   *
FROM     Product
WHERE    category=‘Gadgets’
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

SELECT clause (in practice)

Product

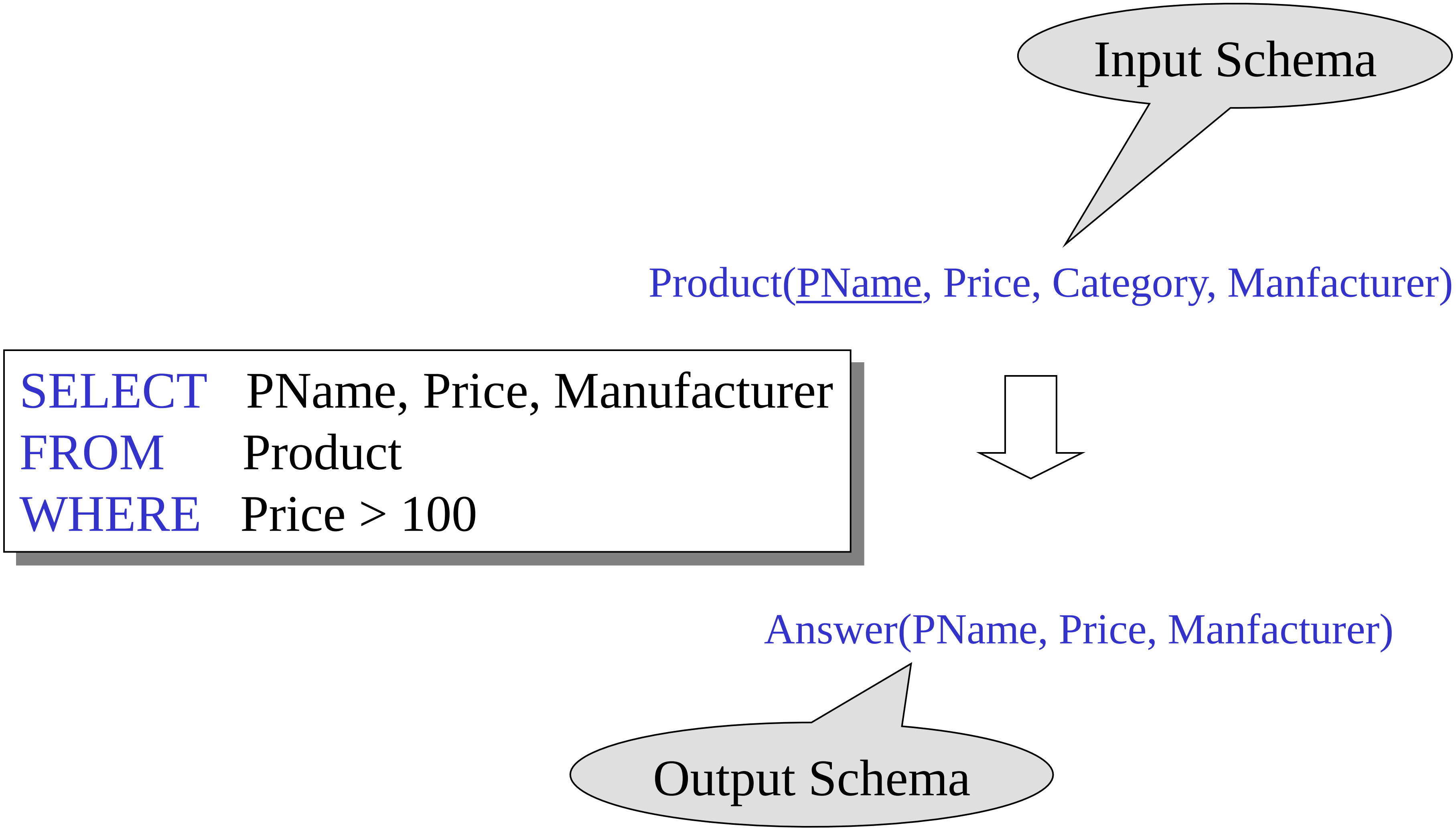
PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

“selection” and “projection”

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```

PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

NOTATION



Product(PName, Price, Category, Manfacturer)

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```

Answer(PName, Price, Manfacturer)

DETAILS

- Case insensitive:
  - Same: SELECT Select select
  - Same: Product   product
  - Different: ‘Seattle’ ‘seattle’

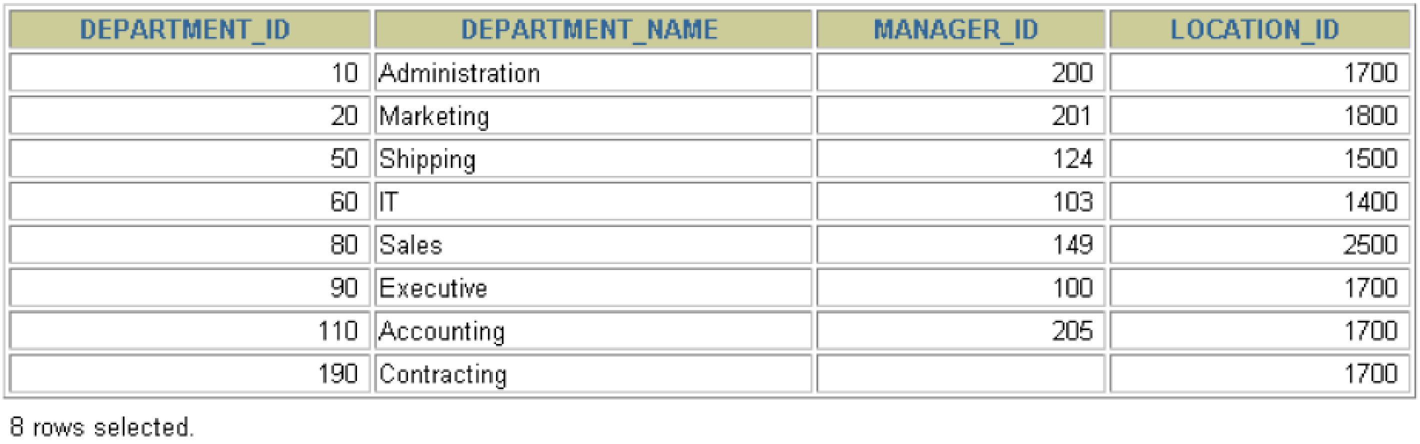
- Constants:
  - ‘abc’ - yes
  - “abc” – no

- Each clause starts in a new line

- Long statements: Split up in separate indented lines

Example:

```
SELECT *
FROM   departments;
```

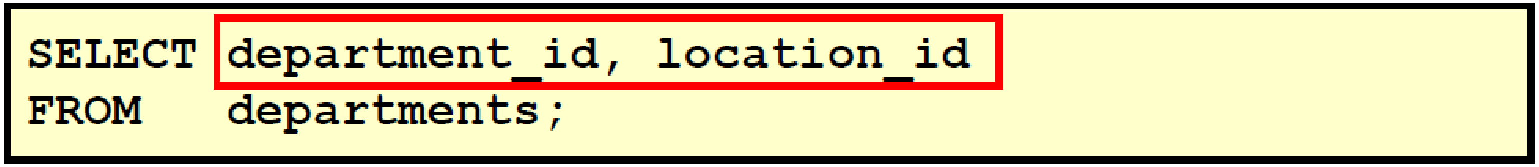


DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

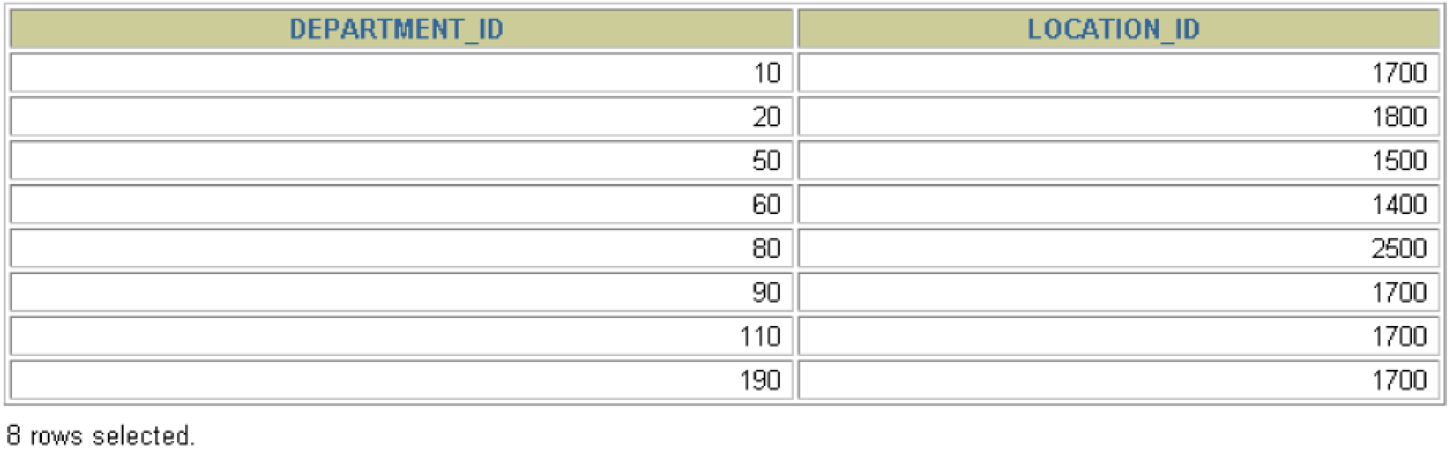
```
8 rows selected.
```

Use SELECT to display ALL columns of the relation (tables).
Keyword = *

Example:



```
SELECT department_id, location_id
FROM   departments;
```



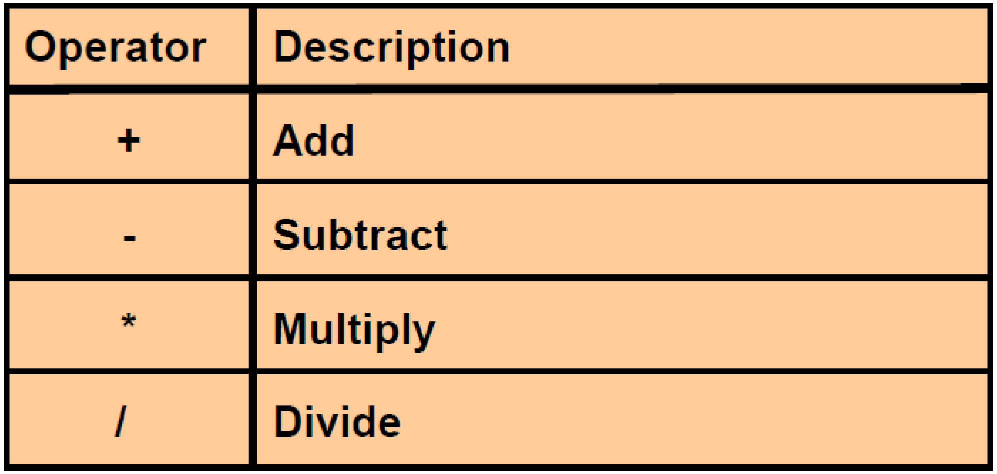
DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

Use SELECT to display specific columns of the relation (tables). Specify the column names separated by commas

Arithmetic Expressions – Add New columns

Create expressions on NUMBER and DATE data with arithmetic operators:

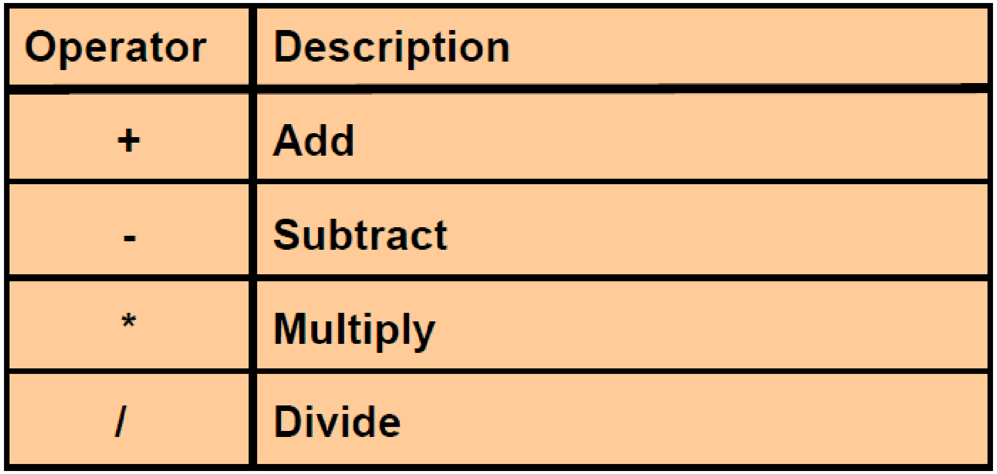


Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Use arithmetic operators in any clause of a SQL statement except the FROM clause.

Arithmetic Expressions – Add New columns

PROPERTIES:

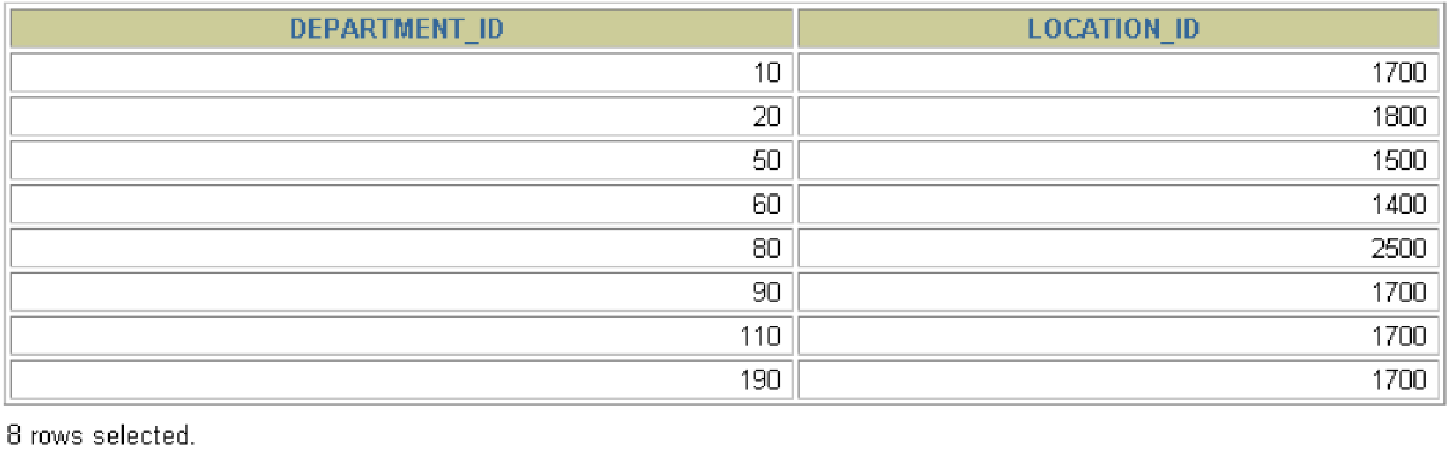


Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

- Multiplication and division take priority over addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to force prioritised evaluation and to clarify statements.

Example:

```
SELECT department_id, location_id
FROM   departments;
```



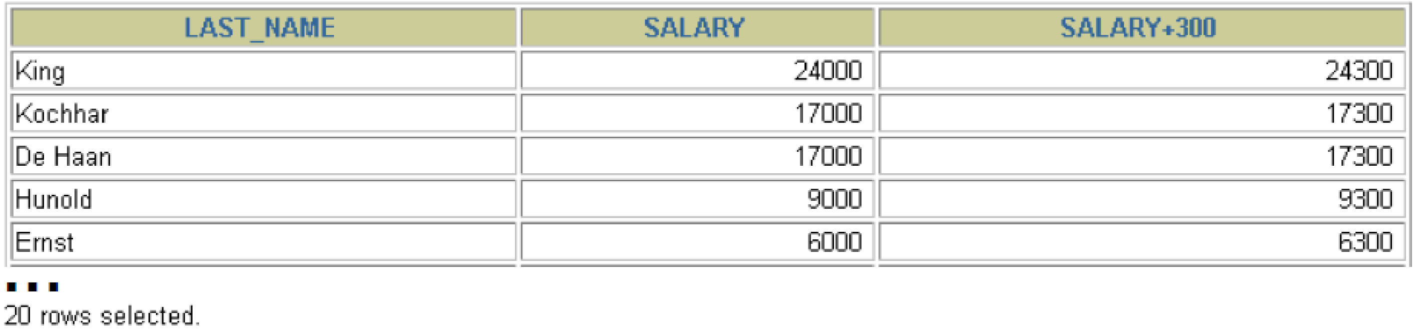
DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

Use SELECT to display specific columns of the relation (tables). Specify the column names separated by commas

Example:

```
SELECT last_name, salary, salary + 300
FROM   employees;
```



LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

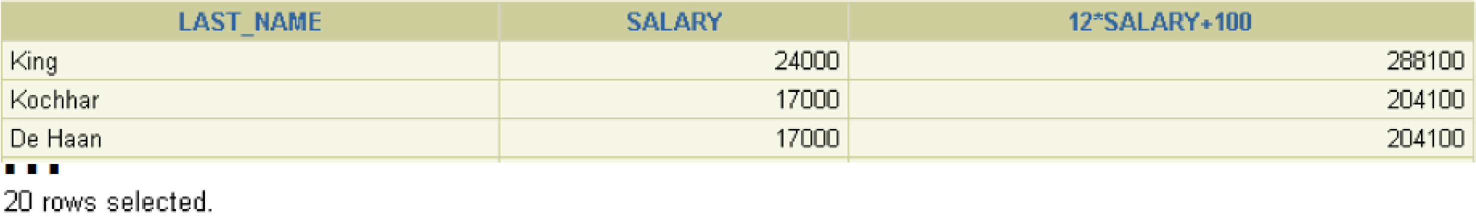
...

20 rows selected.

Example (operator precendence):

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

1



LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100

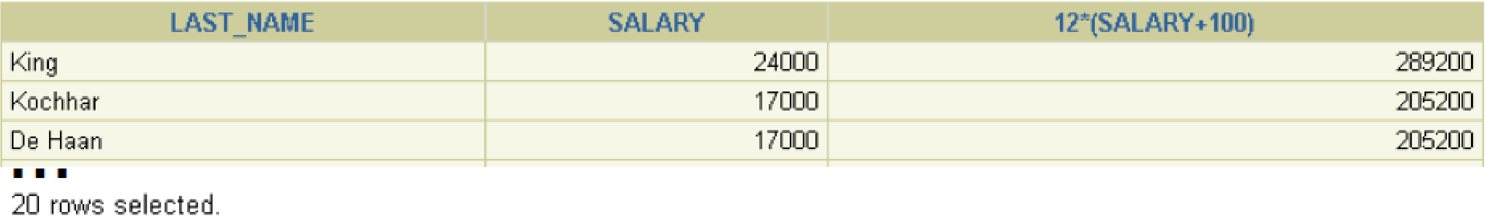
...

20 rows selected.

Using Parentheses

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

2



LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200

...

20 rows selected.

NULL operator in SQL

- Whenever we don’t have a value, we can put a NULL
- Can mean many things:
  - Value does not exists
  - Value exists but is unknown
  - Value not applicable
  - Etc.

- The schema specifies for each attribute if can be null (*nullable* attribute) or not

NULL operator in SQL

- How does SQL cope with tables that have NULLs ?

ANSWER:

If x= NULL then 4*(3-x)/7 is still NULL

If x= NULL then x="Joe" is UNKNOWN

In SQL there are three boolean values:

FALSE	=	0
UNKNOWN	=	0.5
TRUE	=	1

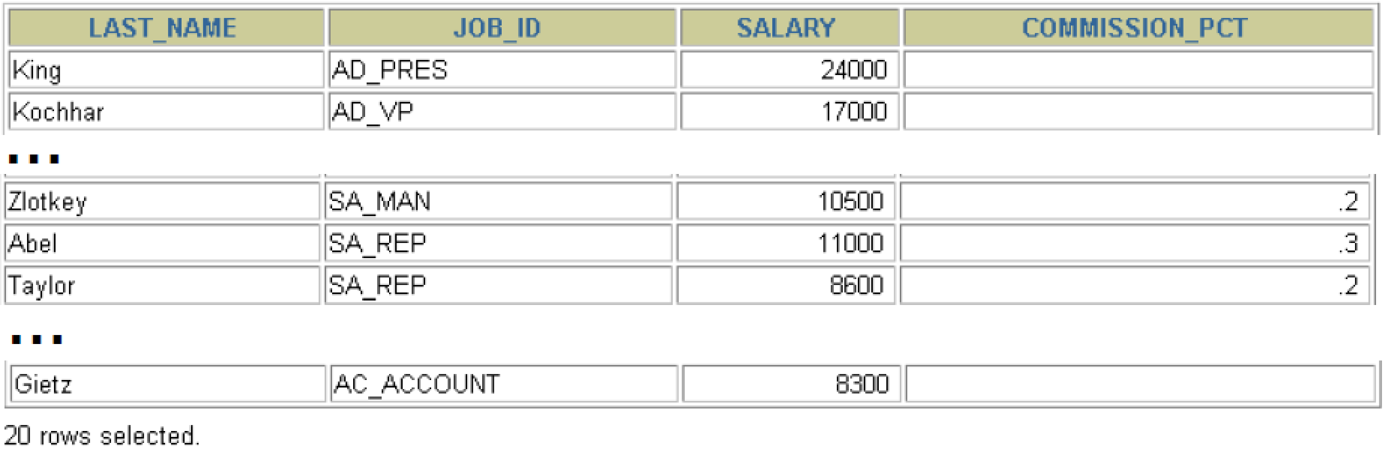
More... next lecture

Example - NULL operator

– A null is a value that is to unavailable, unassigned, unknown, or inapplicable data

– A null is not the same as a zero or a blank space.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

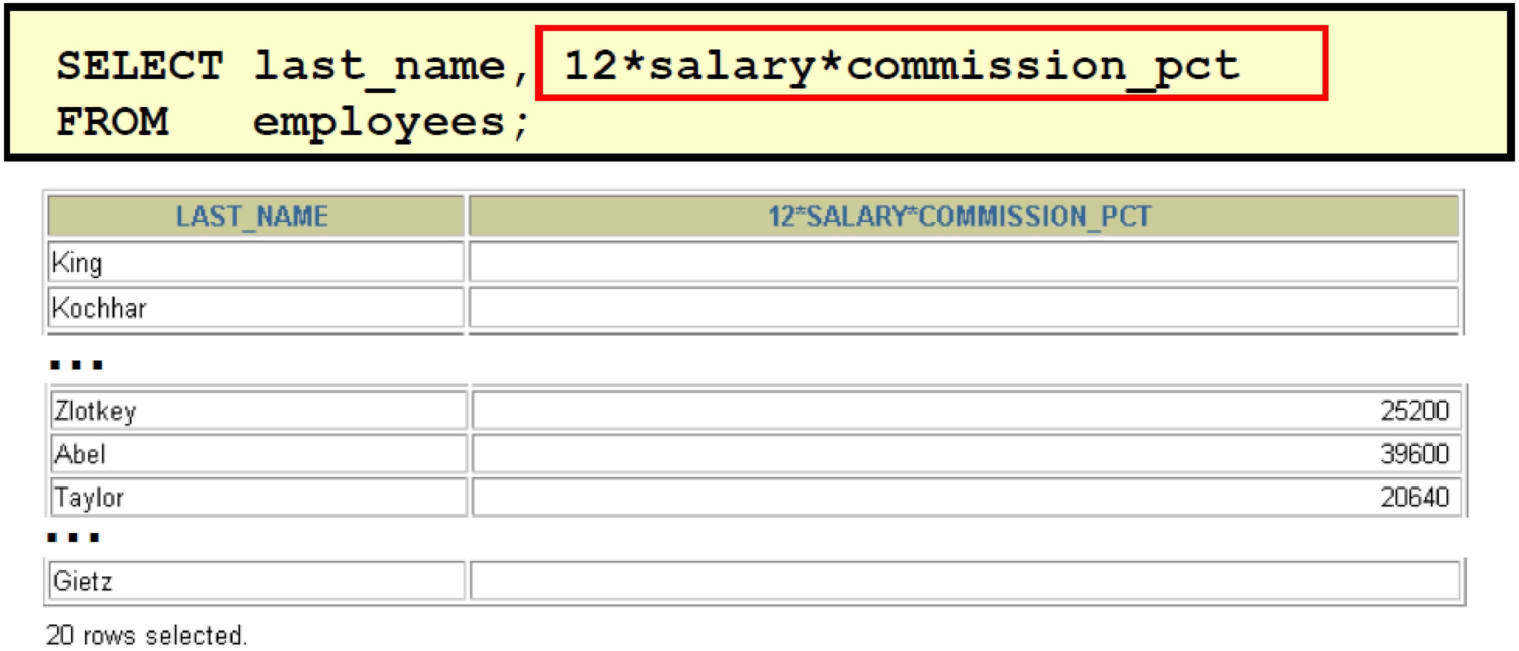


LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

Example - NULL operator

– Arithmetic Operations containing NULL result in NULL.



```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	

...

Zlotkey	25200
Abel	39600
Taylor	20640

...

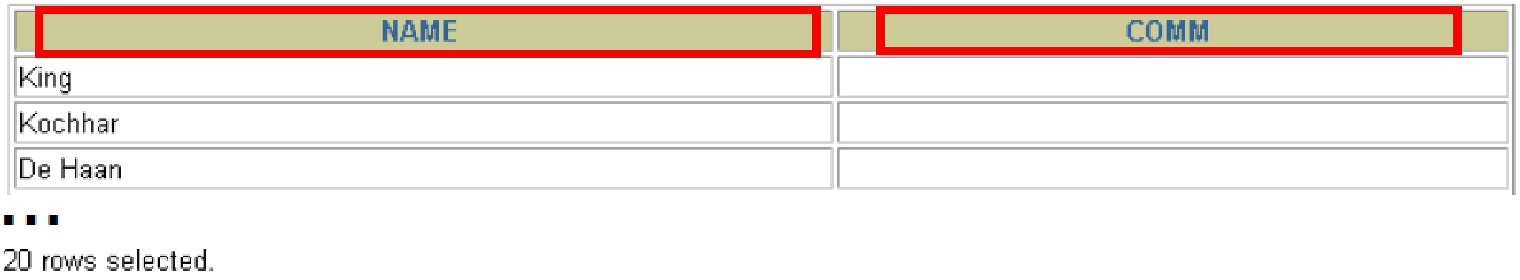
Gietz	

20 rows selected.

Column Alias

A column alias 'renames' a column heading

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```



NAME	COMM
King	
Kochhar	
De Haan	

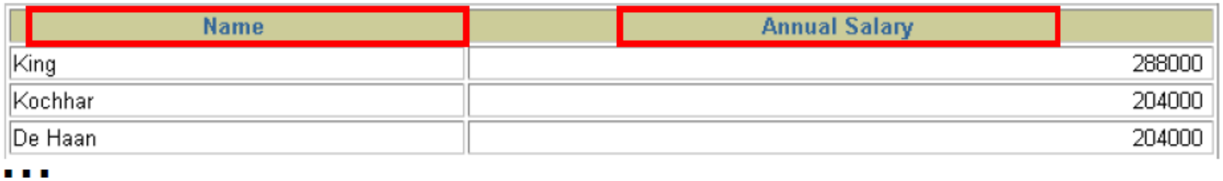
```
...
20 rows selected.
```

Column Alias

A column alias 'renames' a column heading

```
SELECT last_name "Name" , salary*12 "Annual Salary"
FROM   employees;
```

If the alias has a special character (e.g., $, blank space), then it has to be enclosed within double quotation marks.



Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

...

Requires double quotation marks if it contains spaces or special characters or is case sensitive.

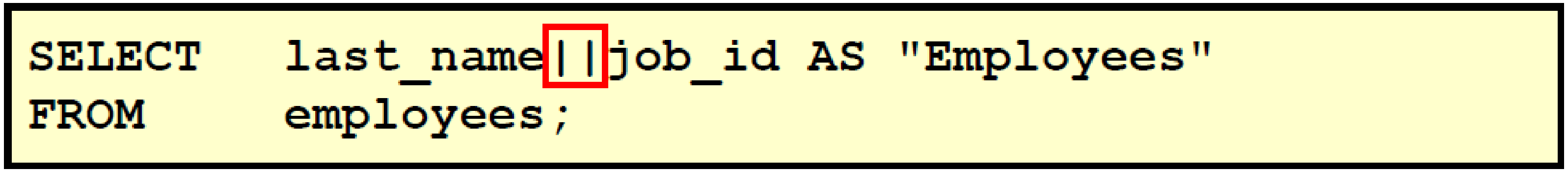
Concatenation Operator

- SQL supports a variety of string operations such as
  - concatenation (using “||”)
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, literal character strings, etc.

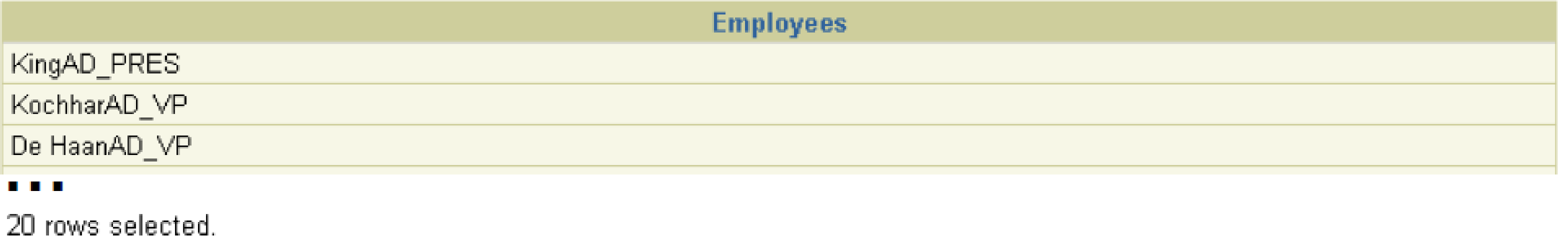
Creates a new column

Concatenation Operator

- SQL supports a variety of string operations such as
  - concatenation (using “||”)
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, literal character strings, etc.



```
SELECT    last_name||job_id AS "Employees"
FROM      employees;
```



Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP

...

20 rows selected.

Creates a new column

Literal Character Strings (LCS)

- SQL supports a variety of string operations such as
  - concatenation (using “||”)
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, literal character strings, etc.

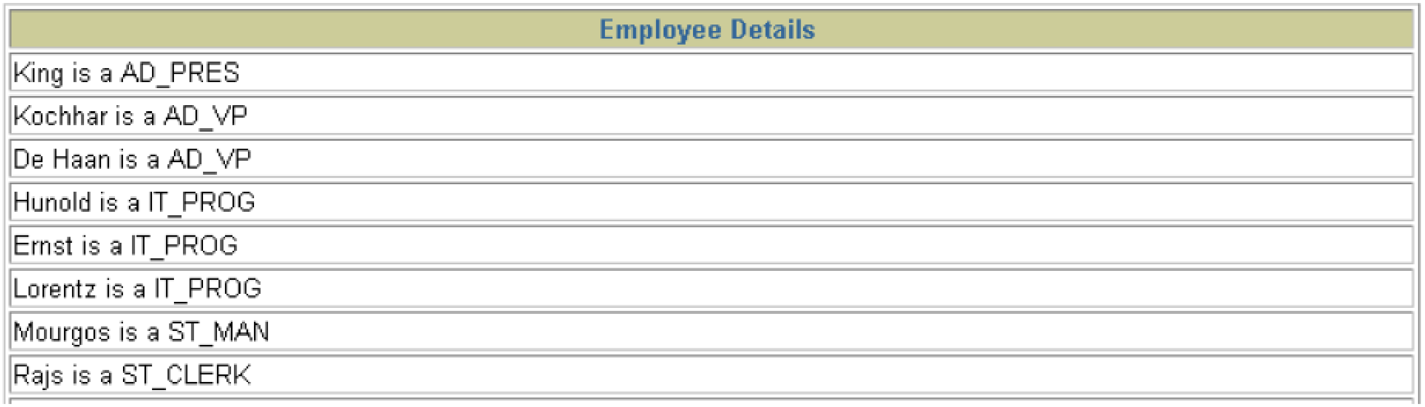
An LCS is any character, expression, or number included in the SELECT list added to the column name or a column alias.

In your SLQ code, this string must be enclosed within single quotations marks

Example:

- SQL supports a variety of string operations such as
  - concatenation (using “||”)
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, literal character strings, etc.

```
SELECT last_name ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```



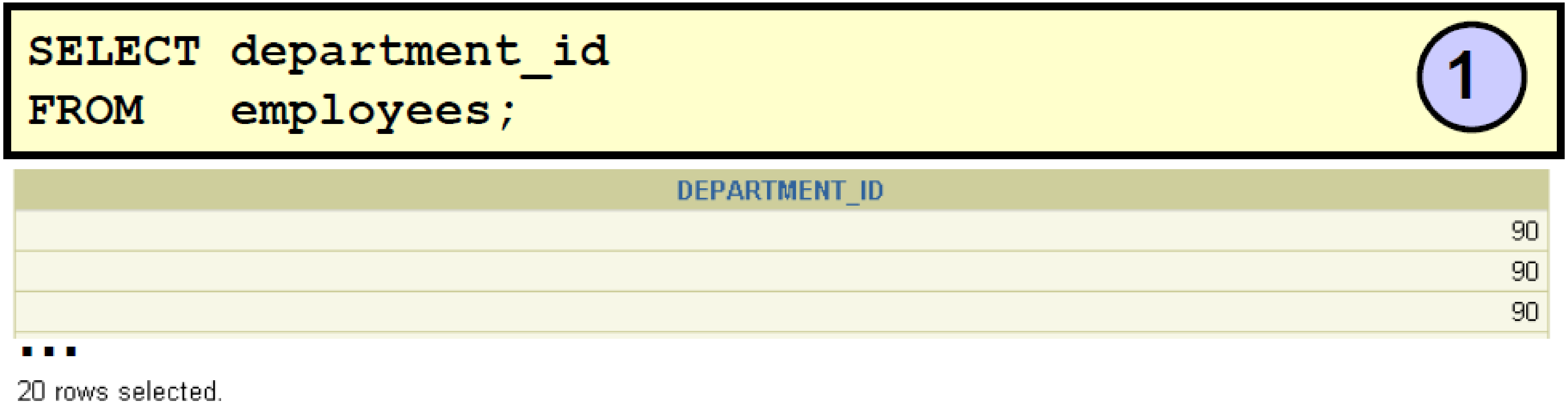
Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

...

IMPORTANT

(a) Duplicate rows

The default display of queries includes duplicate rows (all)



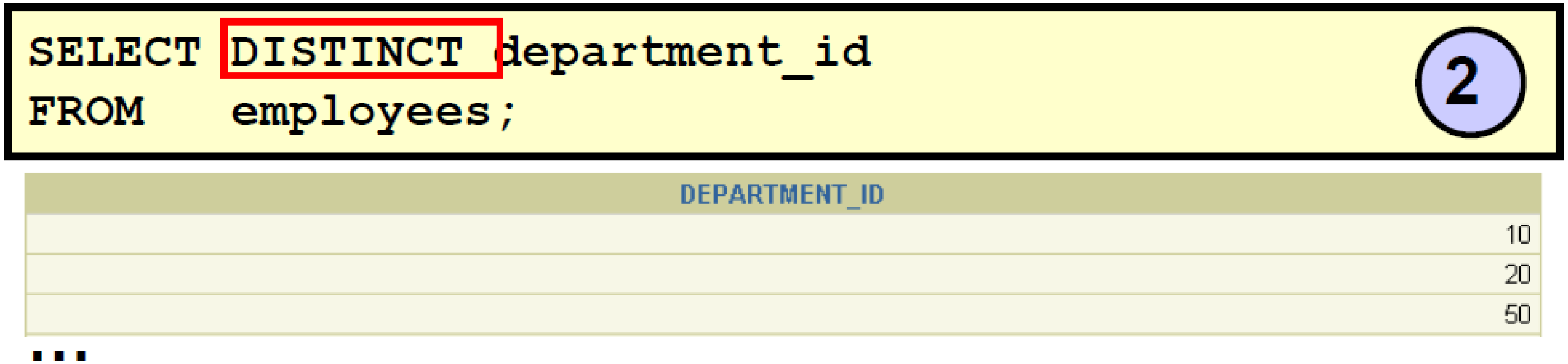
```
SELECT department_id
FROM   employees;
```

```
DEPARTMENT_ID
           90
           90
           90
...
20 rows selected.
```

IMPORTANT

(a) Duplicate rows

The SELECT *DISTINCT* statement is used to return only *distinct* (different) values



```
SELECT DISTINCT department_id
FROM   employees;
```

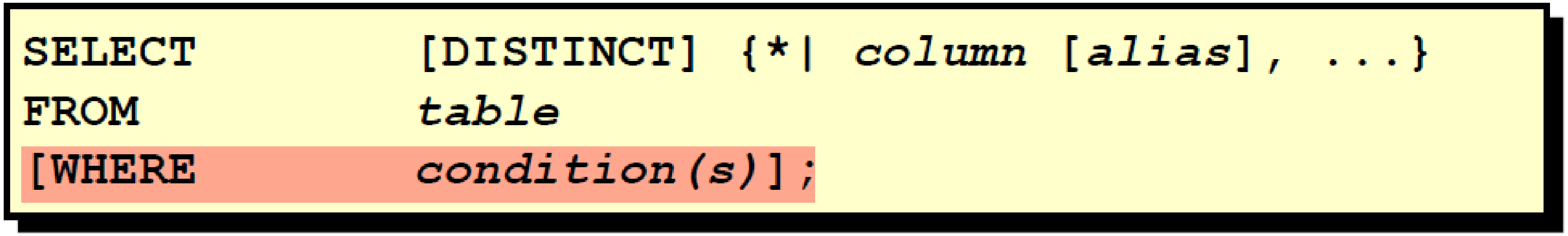
DEPARTMENT_ID
10
20
50

...

The WHERE clause

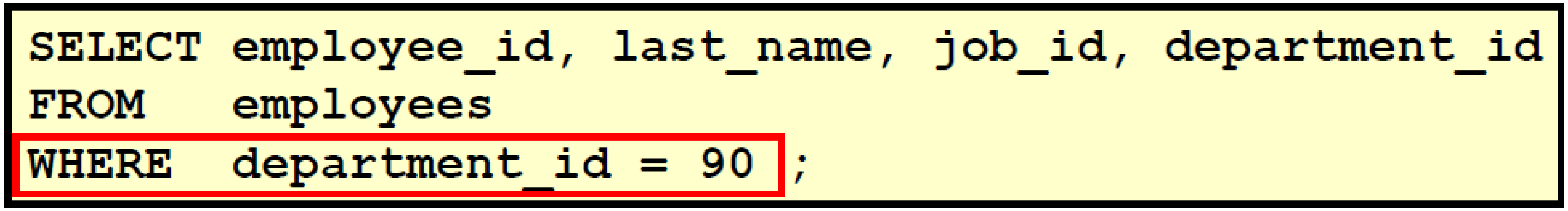
Add restrictions to tuple-selection with the WHERE clause

This statement follows the FROM clause and then ‘;’

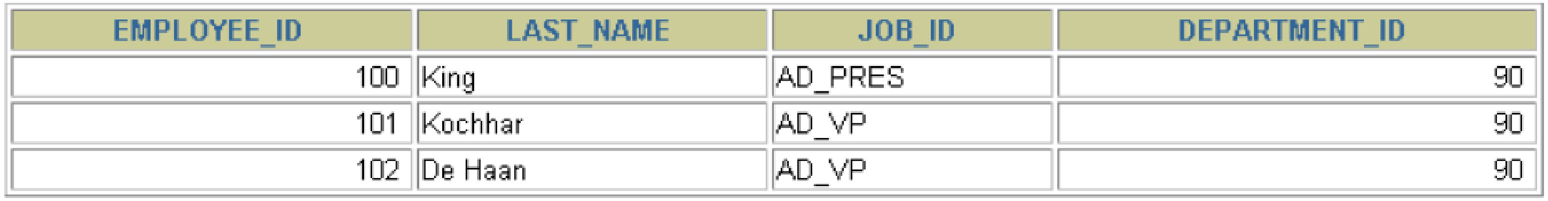


```
SELECT          [DISTINCT] {*| column [alias], ...}
FROM            table
[WHERE          condition(s)];
```

Example:

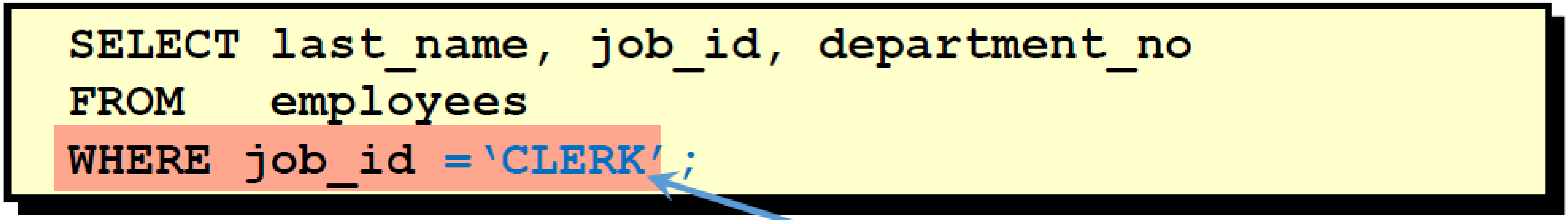


```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```



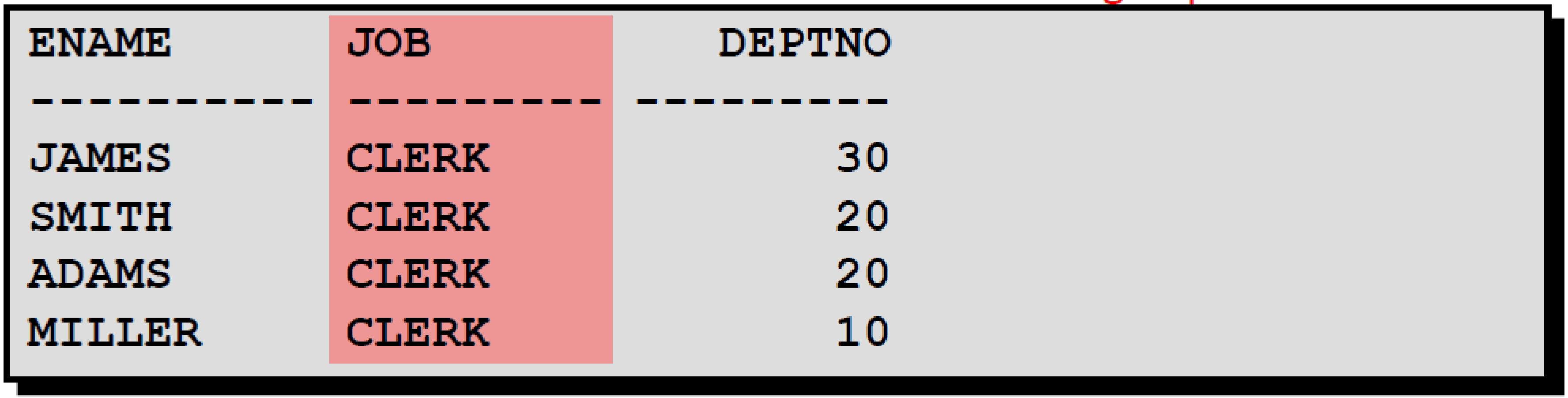
EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

Example:



```
SELECT last_name, job_id, department_no
FROM   employees
WHERE job_id = ‘CLERK’ ;
```

Character strings enclosed within single quotation marks



```
ENAME       JOB            DEPTNO
----------- ---------- ----------
JAMES       CLERK              30
SMITH       CLERK              20
ADAMS       CLERK              20
MILLER      CLERK              10
```

More… next lecture.

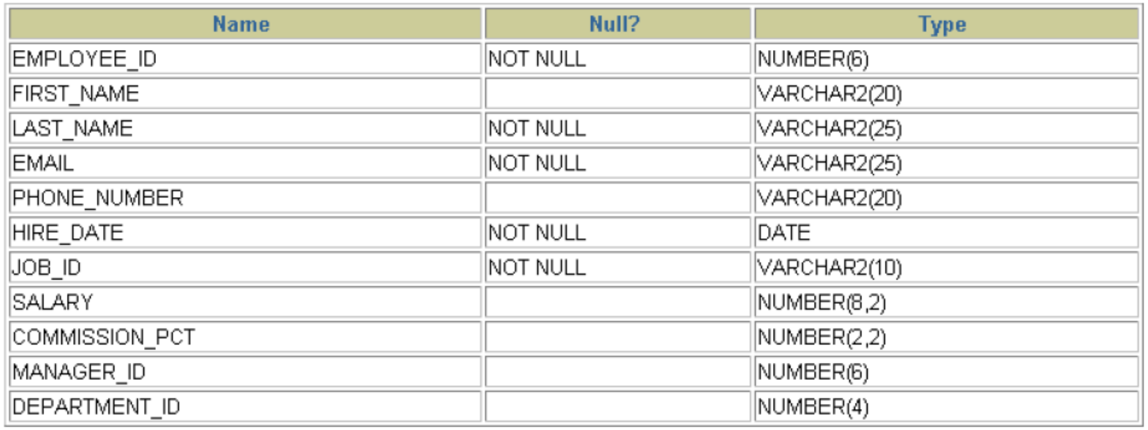
Display Table Structure

Use DESCRIBE to display the internal structure of a table

```
DESC[RIBE] tablename
```

Example:

```
DESCRIBE employees
```



Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Any questions?

Readings: See CANVAS

References:

(a) A Conceptual Poverty Mapping Data Model
Link: https://www.researchgate.net/figure/Key-thematic-layers-for-poverty-spatial-data-modeling_fig2_229724703

(b) Relational Database relationships
https://www.youtube.com/watch?v=C3icLzBtg8I

(c) https://courses.ischool.berkeley.edu/i202/f97/Lecture13/DatabaseDesign/sld002.htm

(d) https://nexwebsites.com/database/database-management-systems/

(e) Acknowledgement – Thanks to http://courses.cs.washington.edu/courses/cse544/ for providing part of this presentation.

(f) Acknowledgement – Thanks to © Silberchatz, Korth and Surdashan for providing part of this presentation.

(e) Malinowski, Elzbieta, Zimányi, Esteban (2008) *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications .* Springer Berlin Heidelberg. Copyright © 2008 Elzbieta Malinowski & Esteban Zimányi