Database Modeling

Mr. Velar Hikmat Elias

Chapter 1

- File system and Database

Introducing the database

- Data vs information
  - Data constitute building blocks of information.
  - Information produced by processing data.
  - Information reveals the meaning of data.
  - Good, timely, relevant information key to decision making.
  - Good decision-making key to organizational survival.

Introduction (Data Model)

- A data model is a conceptual representation of the data structures that are required by a database.
- The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects.
- There are two major methodologies used to create a data model: the Entity-Relationship (ER) approach and the Object Model.

Purpose of database

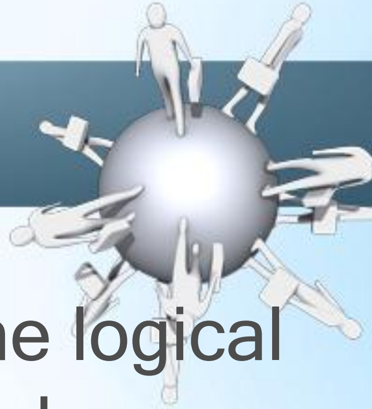
- The purpose of a database
  - To store data
  - To provide an organizational structure for data
  - To provide a mechanism for querying, creating modifying, and deleting data

- A database can store information and relationships that are more complicated than a simple list

Purpose of database

- Create
- Read
- Update
- Delete

Database Design



- Database design is defined as: "design the logical and physical structure of one or more databases to accommodate the information needs of the users in an organization for a defined set of applications". The design process roughly follows five steps:
- 1. planning and analysis
- 2. conceptual design
- 3. logical design
- 4. physical design
- 5. implementation

1. planning and analysis

- Collect all requirements and analyze it
  - Data requirements
  - Function requirements

by using dataflow diagram ,sequences diagram .....etc

2. conceptual design

- After collecting all data, the designer start to design a conceptual schema for the database using (High-level conceptual data Models) like ER-model
- And the conceptual design is a feedback of all data collection .

3. logical design

- Logical design or DATA MODEL MAPPING
- In this case we convert conceptual design from High level conceptual data Model to Implementation data Model
- Like converting E-R model mapping to Relational Model.

4. physical design

- This level shows all saved details by indicating Access paths, file organization for database file.
- Then design the internal schema for DBMS

The Entity-Relationship Model

- The Entity-Relation Model (ER) is the most common method used to build data models for relational databases.
- The ER model views the real world as a construct of entities and association between entities.

1. Entity
2. Attributes
3. Table
4. Coordinate
5. Order
6. Keys
7. Domain
8. NF

Entities

- Entities are the principal data object about which information is to be collected.
- Thing In real world with an Independent existence
- Entities can be classified into:
  1. Physical existence Entities
     - Car, house, person, student
  2. Conceptual existence Entities
     - Job, company, course

Attributes

- Attributes are the properties to describe the Entities.

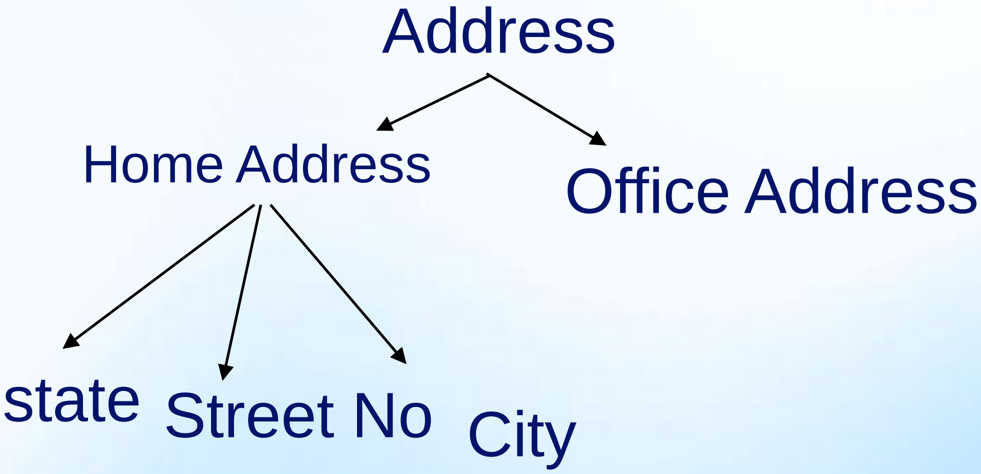
➤ Example:

  - ➤ Entity like “**Employee**”
  - ➤ Have many Attributes like “**Name, SSN ,Gender, salary , address , age** ”

❖ (Attribute Values become a major part of data in the DB)

Attributes tables

- Atomic Attribute (or simple): if it does not contain any meaningful smaller.
- A Composite attribute has multiple components, each of which is atomic or composite, which are sub-part attributes.



Address

Home Address

Office Address

state

Street No

City

Atomic Attribute

- An attribute is considered atomic (or simple) if it does not contain any meaningful smaller components.
- For example, suppose "Gender" is an attribute in our design. The Gender attribute has a small set of possible values, for example M or F. It is not meaningful to decompose Gender into smaller units, and so we say Gender is a Simple attribute.
- As another example consider an attribute for product price, prodPrice. A sample value for prodPrice is $21.03. Of course, one could decompose prodPrice into two attributes where one attribute represents the dollar component (21), and the other attribute represents the cents component (03), but our assumption here is that such a decomposition is not meaningful to the intended application or system that will make use of it. So we would consider prodPrice to be atomic because it cannot be usefully decomposed into meaningful components.

- Exercise:
- Consider an attribute for the employee's last name, such as empLname. Can this be decomposed into smaller meaningful attributes?

Attributes classification

- **Single Valued Attribute:** Attributes that can have single value at a particular instance of time are called single valued. A person can't have more than one age value. Therefore, age of a person is a single-values attribute.

- **Multivalve attribute:** A multi-valued attribute can have more than one value at one time. For example, an entity CAR have a COLOR attribute that represent one color for some or more than one for another.

Database Terminology

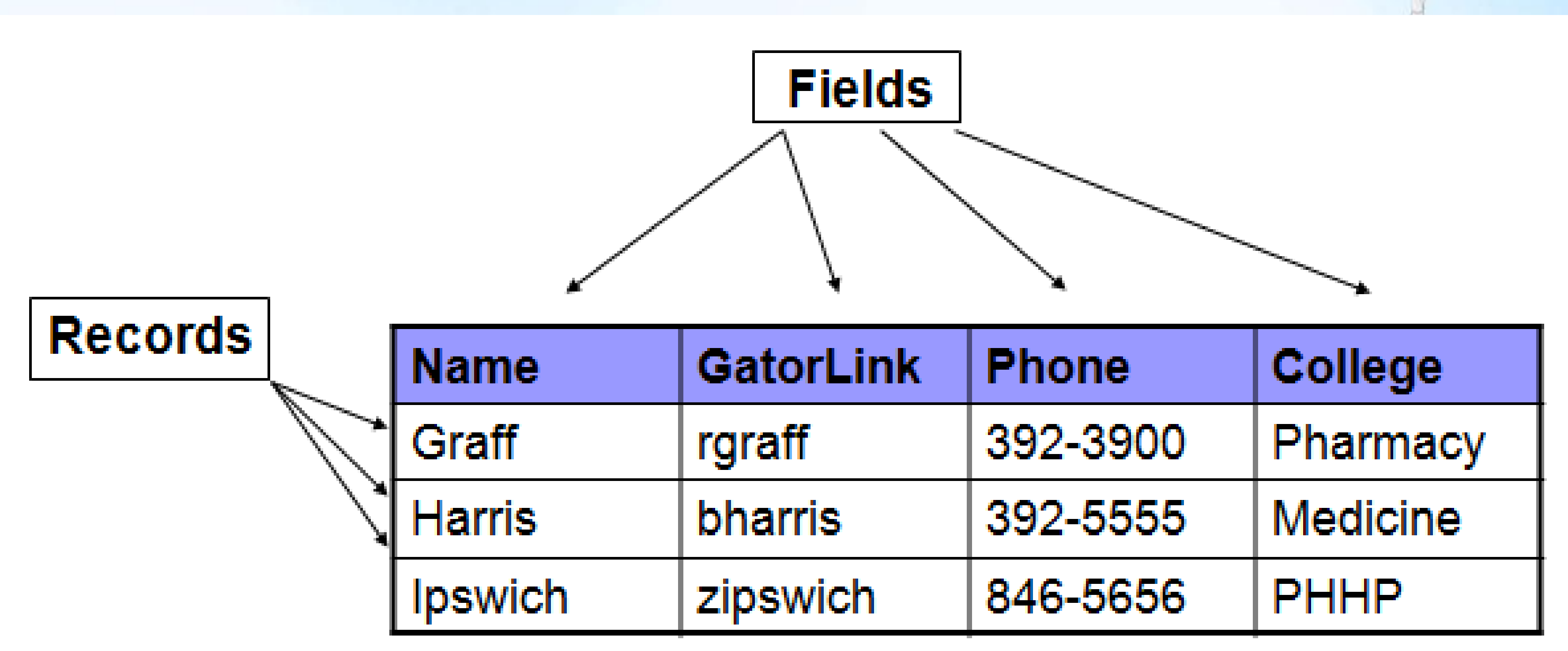
- **Tables** within a relational database hold sets of data using rows and columns
- **Rows** (records) appear horizontally in a report, and contain one or more columns
- **Columns** (fields) are named data elements and appear vertically in a report
- **Primary Keys** identify uniqueness in a row
- **Indexes** are created for faster access to the data in the database

Basic Database Concepts

- Table
  - A set of related records
- Record
  - A collection of data about an individual item
- Field
  - A single item of data common to all records

- **Relation → file → table**
- **Attributes → field → column**
- **Table → record → row**

An Example of a Table

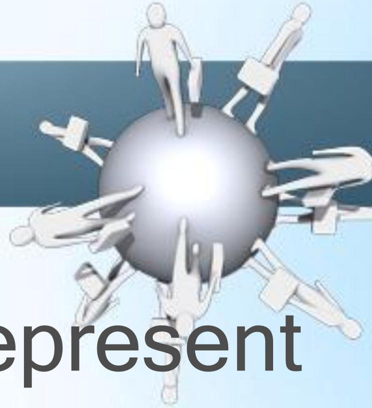


Fields

Records

Name	GatorLink	Phone	College
Graff	rgraff	392-3900	Pharmacy
Harris	bharris	392-5555	Medicine
Ipswich	zipswich	846-5656	PHHP

Null value

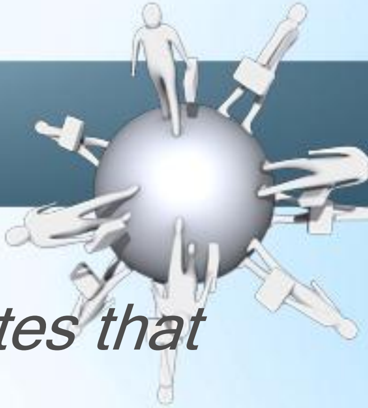


- Some attributes have null value to represent the entity case, and this field is unknown value or not exist.
- Example:
  - College certificate is an attribute to the person entity how complete a college
  - Null value for college certificate is to the person how doesn't have college certificate.

Primary and Foreign Keys

- Primary and foreign keys are the most basic components on which relational theory is based.
- Primary keys enforce entity integrity by uniquely identifying entity instances.
- Foreign keys enforce referential integrity by completing an association between two entities.

Primary key



- The *primary key is an attribute or a set of attributes that uniquely identify a specific instance of* an entity.
- Every entity in the data model must have a primary key whose values uniquely identify instances of the entity.

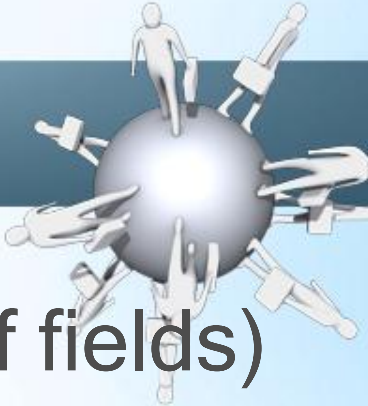
- To qualify as a primary key for an entity, an attribute must have the following properties:
    1. it must have a non-null value for each instance of the entity
    2. the value must be unique (no repeat)for each instance of an entity
    3. the values must not change or become null during the life of each entity instance

In some instances, an entity will have more than one attribute that can serve as a primary key.

Primary Key

- Any key or minimum set of keys that could be a primary key is called a Candidate Key
- Candidate keys which are not chosen as the primary key are known as Alternate keys.
- Sometimes it requires more than one attribute to uniquely identify an entity. A primary key that made up of more than one attribute is known as a *Composite key.*

Foreign Key



- **Foreign key** is a field (or collection of fields) in one table that uniquely identifies a row of another table or the same table.
- In simpler words, the **foreign key** is defined in a second table, but it refers to the primary **key** or a unique **key** in the first table.

Example

Primary Keys

StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

Relation Anomalies

- **Anomalies** are problems that can occur in poorly planned, un-normalised **databases** where all the **data** is stored in one table (a flat-file **database**).
- **Data Anomalies**. Normalization is the process of splitting relations into well structured relations that allow users to insert, delete, and update tables without introducing **database** inconsistencies.
- Without normalization many problems can occur when trying to load an integrated conceptual **model** into the DBMS.

Relation anomalies

Employees table

ID	Name	Salary	Project	P. No.	Project Address
1	Azad	500	A	1001	Hawler
2	Naz	300	B	1002	Koya
3	Kawa	250	A	1001	Hawler
4	Dara	300	C	1003	Karkuk
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1000	.	.	.	.	.

77B

For Each row

2B 30B 5B 2B 8B 30B

Relation anomalies

- Each cell has different storage capacity
- row capacity = ∑ of all cells for same row
- row capacity = 2+30+5+2+8+30 = 77 Byte

- If the table have 1000 rows
- Then table capacity = 1000 x 77B = 77000B

- This is very big capacity if we have huge database system

Relation anomalies

- Database modeling required ?
- Start split the table to the two Relational tables
- With indicate the Key for these tables

Relation anomalies

Employees table-1

Table capacity
= 1000 x 39B
= 39000 Byte

ID	Name	Salary	Project
1	Azad	500	A
2	Naz	300	B
3	Kawa	250	A
4	Dara	300	C
.	.	.	.
.	.	.	.
.	.	.	.
1000			

← 39B For Each row

2B 30B 5B 2B

Relation anomalies

Employees table-2

Project	P. No.	Project Address
A	1001	Hawler
B	1002	Koya
C	1003	Karkuk

2B 8B 30B

40B For Each row

Table capacity
= 3 x 40B
= 120 Byte

And the total capacity for two tables is
39000 B + 120 B = 39120 Byte

Relation anomalies

- To complete the model → we need a common column link between two tables
- Some time we need to split the tables to protect the database system from loss the information

Relation anomalies

ID	Name	Activity	Fees
001	Azad	Football	50 $
002	Aree	Golf	40 $
001	Azad	Basketball	30 $
003	Naz	Swimming	20 $
002	Aree	Swimming	20 $
004	Khalid	Football	50 $

If we remove Aree from Golf

We remove Golf Fees (40 $) with it
And the system will lose this value

To remove this anomalies we need to split the table to Two tables

Relation anomalies

Functional dependency

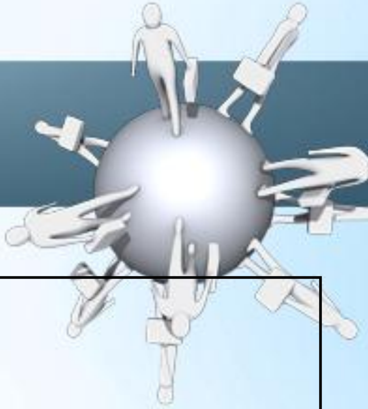
ID	Name	Activity
001	Azad	Football
002	Aree	Golf
001	Azad	Basketball
003	Naz	Swimming
002	Aree	Swimming
004	Khalid	Football

Activity	Fees
Football	50 $
Golf	40 $
Basketball	30 $
Swimming	20 $

Functional dependency is the link between two tables

Important to indicating the KEYS

Relation anomalies



- X → Y (X determines Y)
- Or
- Y functionally dependency to X

- Ex:
  - customer account → customer balance

determine

Functional dependency

  - Ex: (X,Z) → Y X → (Y,Z)

Normalization

- Normalization is a design technique that is widely used as a guide in designing relational
- databases.
- Normalization is essentially a two step process that puts data into tabular form by removing repeating groups and then removes duplicated from the relational tables.
- Normalization theory is based on the concepts of **normal forms.**

Normalization

- Normalization generally involves splitting existing table into multiple ones, which must be re-joined or linked each time a query is issued

- Steps of Normalization
  1. First Normal Form (1NF)
  2. Second Normal Form (2NF)
  3. Third Normal Form (3NF)

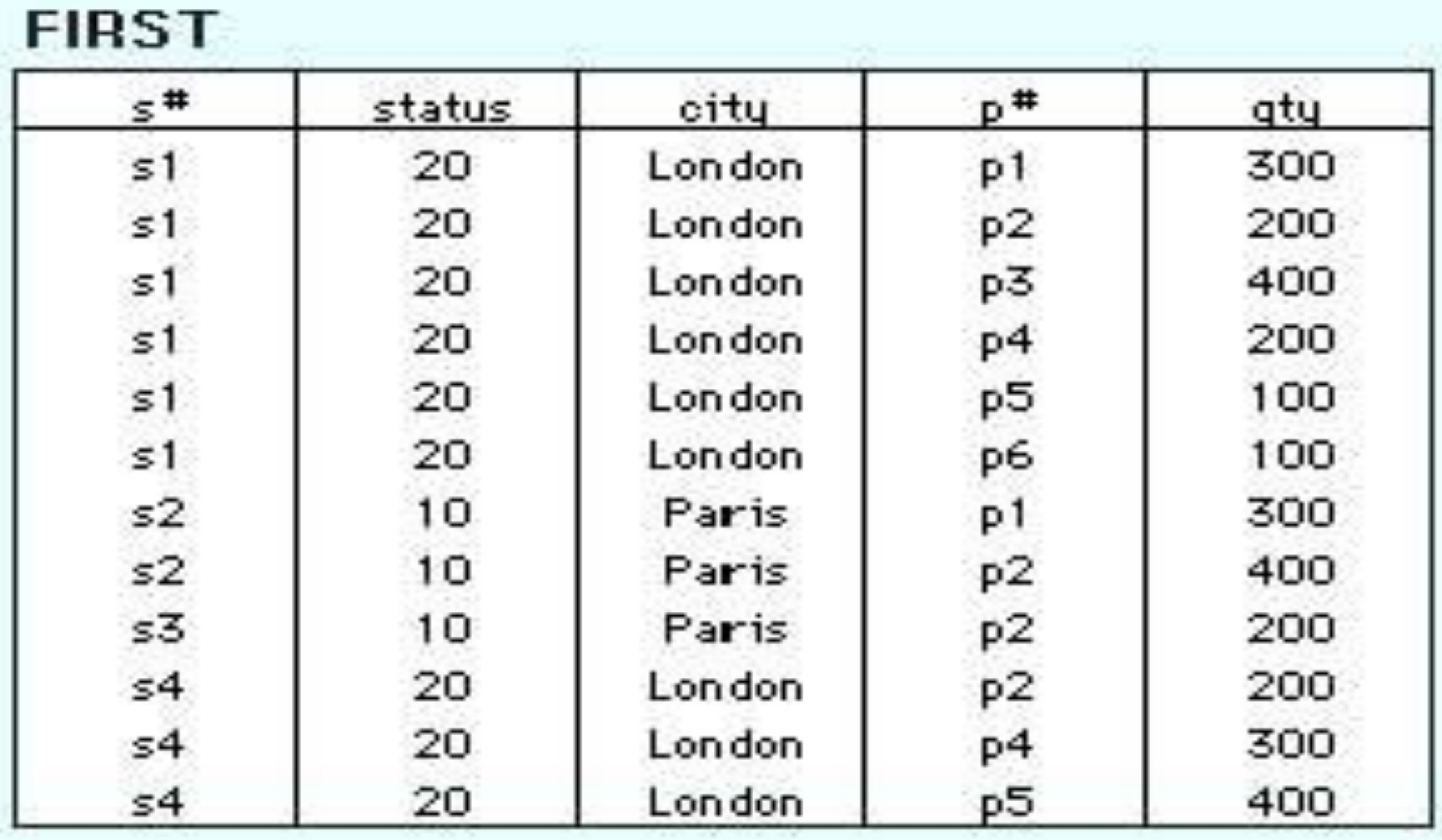
In practice these normal form are enough for good database design.

Normalization-1NF

- A relational table, by definition, is in first normal form. All values of the columns are atomic. That is, they contain no repeating values. Figure1 shows the table (Supplier) in 1NF.

- Although the table FIRST is in 1NF it contains redundant data.
- For example, information about the supplier's location and the location's status have to be repeated for every part supplied.

Normalization-1NF



FIRST

s#	status	city	p#	qty
s1	20	London	p1	300
s1	20	London	p2	200
s1	20	London	p3	400
s1	20	London	p4	200
s1	20	London	p5	100
s1	20	London	p6	100
s2	10	Paris	p1	300
s2	10	Paris	p2	400
s3	10	Paris	p2	200
s4	20	London	p2	200
s4	20	London	p4	300
s4	20	London	p5	400

Normalization-1NF

- **s# -** supplier identification number (this is the primary key)
- **Status -** status code assigned to city
- **city -** name of city where supplier is located
- **p#** - part number of part supplied
- **Qty -** quantity of parts supplied to date

Normalization-1NF

- Redundancy causes what are called *update anomalies*. *Update anomalies are problems that arise* when information is inserted, deleted, or updated. For example, the following anomalies could occur in FIRST:

1. INSERT. The fact that a certain supplier (s5) is located in a particular city (Athens) cannot be added until they supplied a part.
2. DELETE. If a row is deleted, then not only is the information about quantity and part lost but also information about the supplier.
3. UPDATE. If supplier s1 moved from London to New York, then six rows would have to be updated with this new information.

Normalization-2NF

- The definition of second normal form states that only tables with composite primary keys can be in 1NF but not in 2NF.
- A relational table is in second normal form 2NF if it is in 1NF and every non-key column is fully dependent upon the primary key.
- All partial dependencies are removed to place in another table

Normalization-2NF

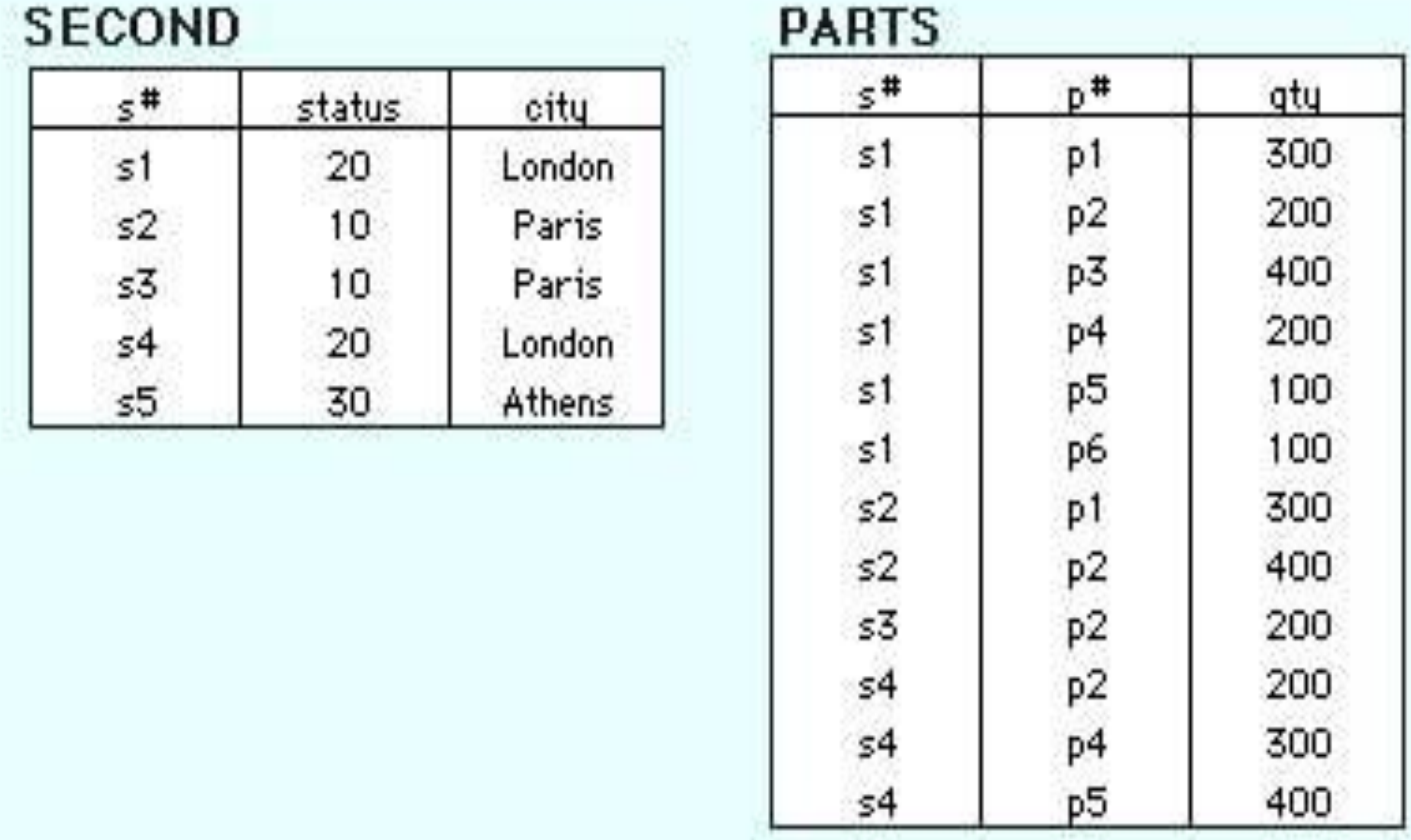
- Table “FIRST” is in 1NF but not in 2NF because status and city are functionally dependent upon only on the column S# of the composite key (S#, P#).

- S# → city, status
- City → status
- (S#, P#) → qty

Normalization-2NF

- To transform FIRST into 2NF we move the columns s#, status, and city to a new table called SECOND. The column s# becomes the primary key of this new table.

Normalization-2NF



SECOND

s#	status	city
s1	20	London
s2	10	Paris
s3	10	Paris
s4	20	London
s5	30	Athens

PARTS

s#	p#	qty
s1	p1	300
s1	p2	200
s1	p3	400
s1	p4	200
s1	p5	100
s1	p6	100
s2	p1	300
s2	p2	400
s3	p2	200
s4	p2	200
s4	p4	300
s4	p5	400

Normalization-2NF

- Tables in 2NF but not in 3NF still contain modification anomalies.
- In the example of SECOND, they are:
  - INSERT. The fact that a particular city has a certain status (Rome has a status of 50) cannot be inserted until there is a supplier in the city.

  - DELETE. Deleting any row in SUPPLIER destroys the status information about the city as well as the association between supplier and city.

Normalization-3NF

- The third normal form requires that all columns in a relational table are dependent only upon the primary key.
- A more formal definition is:
  - A relational table is in third normal form (3NF) if it is already in 2NF and every non-key column is non transitively dependent upon its primary key.
  - In other words, all nonkey attributes are functionally dependent only upon the primary key.

Normalization-3NF

- Table PARTS is already in 3NF. The non-key column, qty, is fully dependent upon the primary key (s#, p#).
- SUPPLIER is in 2NF but not in 3NF because it contains a *transitive dependency.*
- transitive dependency is occurs when a non-key column that is a determinant of the primary key is the determinate of other columns.

Normalization-3NF

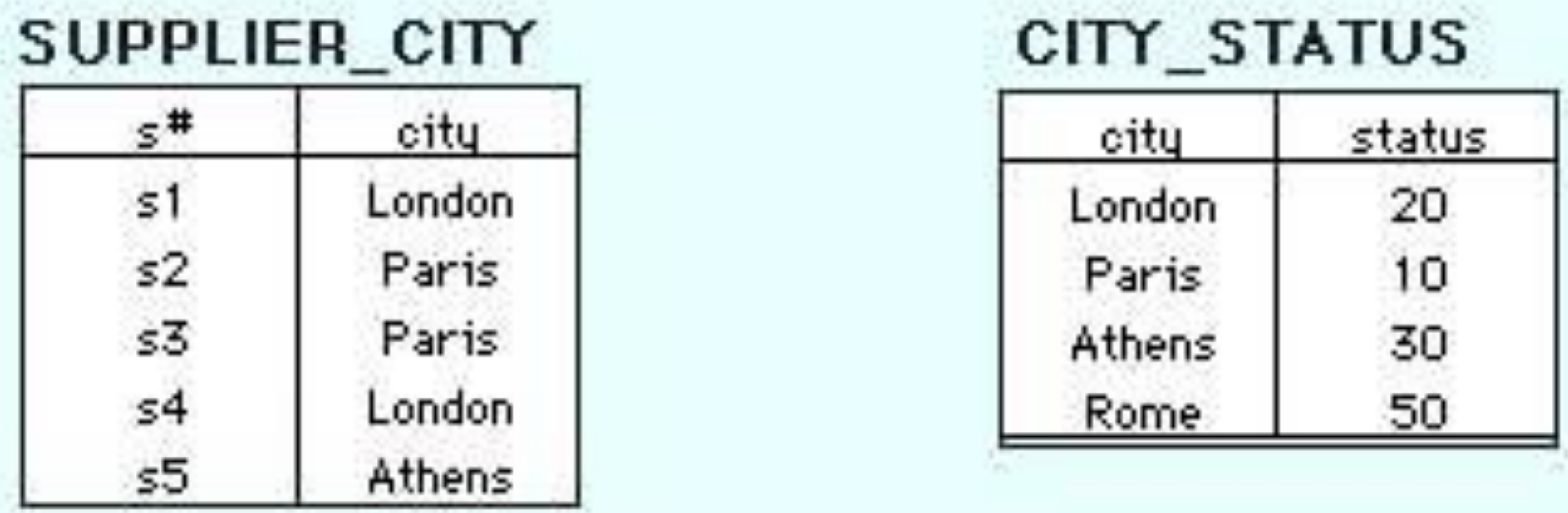
- The concept of a transitive dependency can be illustrated by showing the functional dependencies in SUPPLIER:
  - SUPPLIER.s# –> SUPPLIER.status
  - SUPPLIER.s# –> SUPPLIER.city
  - SUPPLIER.city–> SUPPLIER.status

- Note that SUPPLIER.status is determined both by the primary key s# and the non-key column city.

Normalization-3NF

- To transform SUPPLIER into 3NF,
- We create a new table called CITY_STATUS and move the columns city and status into it.
- Status is deleted from the original table, city is left behind to serve as a foreign key to CITY_STATUS, and the original table is renamed to SUPPLIER_CITY to reflect its semantic meaning.

Normalization-3NF



SUPPLIER_CITY

s#	city
s1	London
s2	Paris
s3	Paris
s4	London
s5	Athens

CITY_STATUS

city	status
London	20
Paris	10
Athens	30
Rome	50

Advantages of Third Normal Form

- The advantages to having relational tables in 3NF is that it eliminates redundant data which in turn saves space and reduces manipulation anomalies. For example, the improvements to our sample database are:
  - INSERT. Facts about the status of a city, Rome has a status of 50, can be added even though there is not supplier in that city. Likewise, facts about new suppliers can be added even though they have not yet supplied parts.
  - DELETE. Information about parts supplied can be deleted without destroying information about a supplier or a city.
  - UPDATE. Changing the location of a supplier or the status of a city requires modifying only one row.

Normalization-3NF

- These can be represented in "psuedo-SQL" as:

- PARTS (#s, p#, qty)
- Primary Key (s#, #p)
- Foreign Key (s#) references SUPPLIER_CITY.s#

- SUPPLIER_CITY(s#, city)
- Primary Key (s#)
- Foreign Key (city) references CITY_STATUS.city

- CITY_STATUS (city, status)
- Primary Key (city)

Chapter 2

SQL AND FILTERS

SQL-Structured Query Language

- SQL: is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS).
- SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

SQL-select

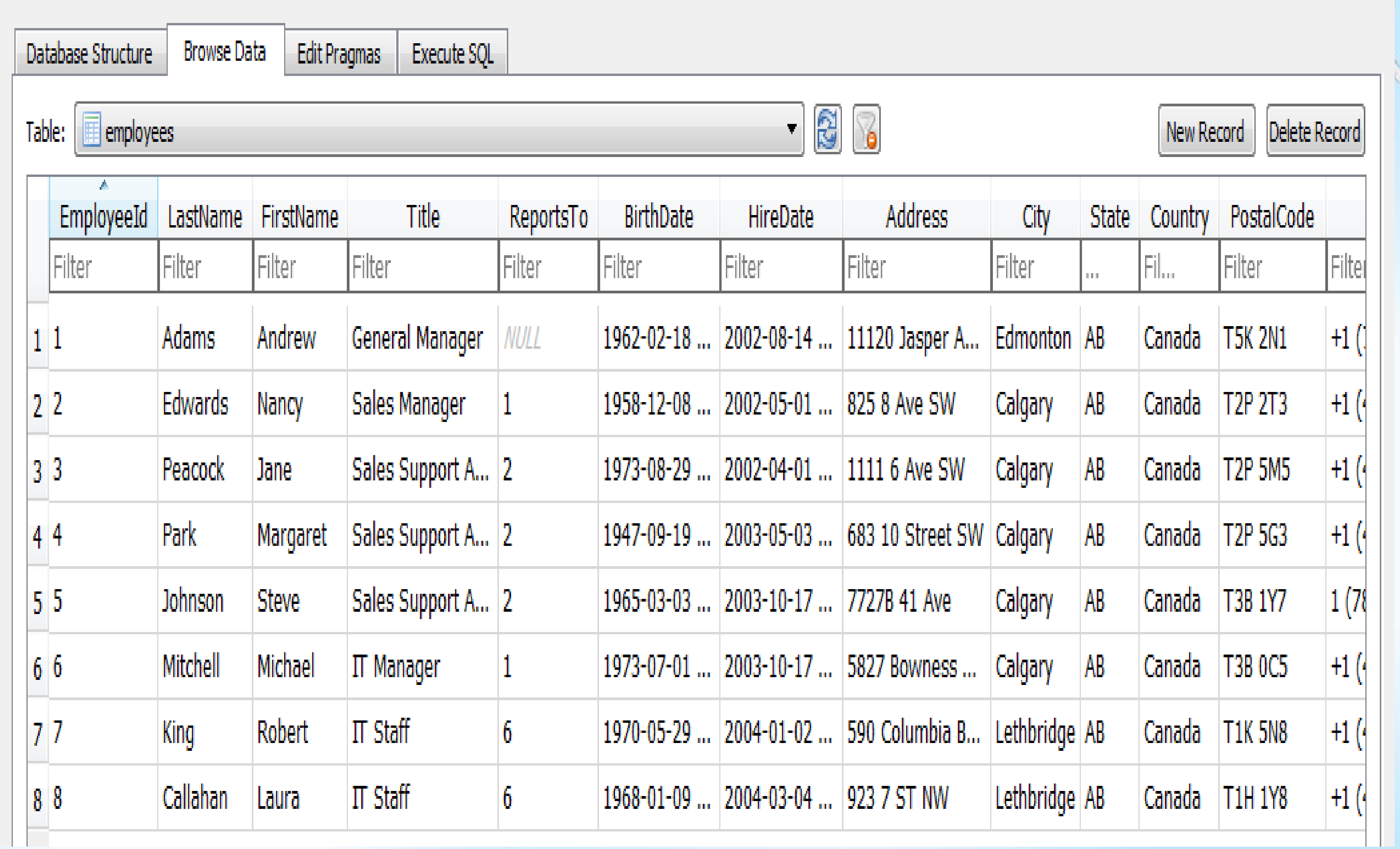


Table: employees

	EmployeeId	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	City	State	Country	PostalCode	
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	...	Fil...	Filter	Filte
1	1	Adams	Andrew	General Manager	NULL	1962-02-18 ...	2002-08-14 ...	11120 Jasper A...	Edmonton	AB	Canada	T5K 2N1	+1 (
2	2	Edwards	Nancy	Sales Manager	1	1958-12-08 ...	2002-05-01 ...	825 8 Ave SW	Calgary	AB	Canada	T2P 2T3	+1 (
3	3	Peacock	Jane	Sales Support A...	2	1973-08-29 ...	2002-04-01 ...	1111 6 Ave SW	Calgary	AB	Canada	T2P 5M5	+1 (
4	4	Park	Margaret	Sales Support A...	2	1947-09-19 ...	2003-05-03 ...	683 10 Street SW	Calgary	AB	Canada	T2P 5G3	+1 (
5	5	Johnson	Steve	Sales Support A...	2	1965-03-03 ...	2003-10-17 ...	7727B 41 Ave	Calgary	AB	Canada	T3B 1Y7	1 (78
6	6	Mitchell	Michael	IT Manager	1	1973-07-01 ...	2003-10-17 ...	5827 Bowness ...	Calgary	AB	Canada	T3B 0C5	+1 (
7	7	King	Robert	IT Staff	6	1970-05-29 ...	2004-01-02 ...	590 Columbia B...	Lethbridge	AB	Canada	T1K 5N8	+1 (
8	8	Callahan	Laura	IT Staff	6	1968-01-09 ...	2004-03-04 ...	923 7 ST NW	Lethbridge	AB	Canada	T1H 1Y8	+1 (

SQL-select

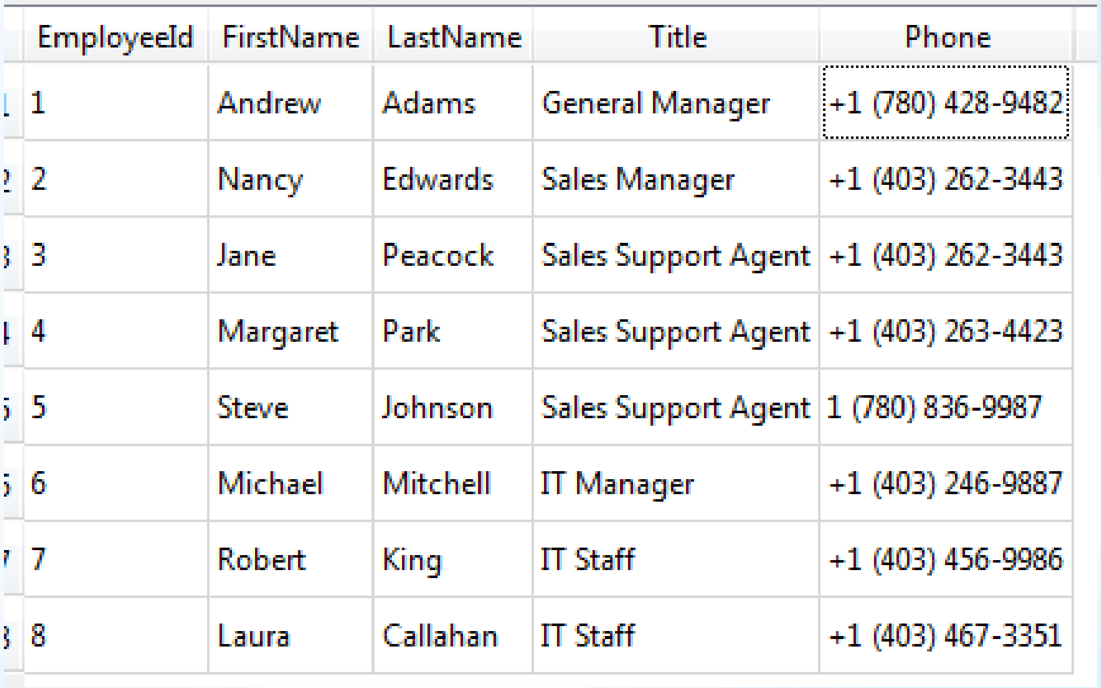
- select *
- from employees

- This command will shows all attributes from table Employees

- Note: make sure from tables name

SQL-select

- select EmployeeId, FirstName, Lastname, Title, Phone
- from employees



EmployeeId	FirstName	LastName	Title	Phone
1	Andrew	Adams	General Manager	+1 (780) 428-9482
2	Nancy	Edwards	Sales Manager	+1 (403) 262-3443
3	Jane	Peacock	Sales Support Agent	+1 (403) 262-3443
4	Margaret	Park	Sales Support Agent	+1 (403) 263-4423
5	Steve	Johnson	Sales Support Agent	1 (780) 836-9987
6	Michael	Mitchell	IT Manager	+1 (403) 246-9887
7	Robert	King	IT Staff	+1 (403) 456-9986
8	Laura	Callahan	IT Staff	+1 (403) 467-3351

SQL-select

- select EmployeeId, phone, lastname, firstname, Phone
- from employees

EmployeeId	Phone	LastName	FirstName	Phone
1	+1 (780) 428-9482	Adams	Andrew	+1 (780) 428-9482
2	+1 (403) 262-3443	Edwards	Nancy	+1 (403) 262-3443
3	+1 (403) 262-3443	Peacock	Jane	+1 (403) 262-3443
4	+1 (403) 263-4423	Park	Margaret	+1 (403) 263-4423
5	1 (780) 836-9987	Johnson	Steve	1 (780) 836-9987
6	+1 (403) 246-9887	Mitchell	Michael	+1 (403) 246-9887
7	+1 (403) 456-9986	King	Robert	+1 (403) 456-9986
8	+1 (403) 467-3351	Callahan	Laura	+1 (403) 467-3351

SQL-select

- select EmployeeId, firstname, lastname, reportsto
- from employees
- order by reportsto

EmployeeId	FirstName	LastName	ReportsTo
1	Andrew	Adams	NULL
2	Nancy	Edwards	1
6	Michael	Mitchell	1
3	Jane	Peacock	2
4	Margaret	Park	2
5	Steve	Johnson	2
7	Robert	King	6
8	Laura	Callahan	6

- Try it out:::::::
- select EmployeeId, firstname, lastname, reportsto
- from employees
- order by firstname

SQL-select

- Sorting multiple columns
- select EmployeeId, FirstName, Lastname, Title, reportsto
- from employees
- Order by firstname and reportsto;

EmployeeId	FirstName	LastName	Title	ReportsTo
1	Andrew	Adams	General Manager	NULL
2	Nancy	Edwards	Sales Manager	1
3	Jane	Peacock	Sales Support Agent	2
4	Margaret	Park	Sales Support Agent	2
5	Steve	Johnson	Sales Support Agent	2
6	Michael	Mitchell	IT Manager	1
7	Robert	King	IT Staff	6
8	Laura	Callahan	IT Staff	6

SQL-select

- Sorting by column position
- select EmployeeId, FirstName, Lastname, Title, reportsto
- from employees
- Order by 2 and 5;

EmployeeId	FirstName	LastName	Title	ReportsTo
1	Andrew	Adams	General Manager	NULL
2	Nancy	Edwards	Sales Manager	1
3	Jane	Peacock	Sales Support Agent	2
4	Margaret	Park	Sales Support Agent	2
5	Steve	Johnson	Sales Support Agent	2
6	Michael	Mitchell	IT Manager	1
7	Robert	King	IT Staff	6
8	Laura	Callahan	IT Staff	6

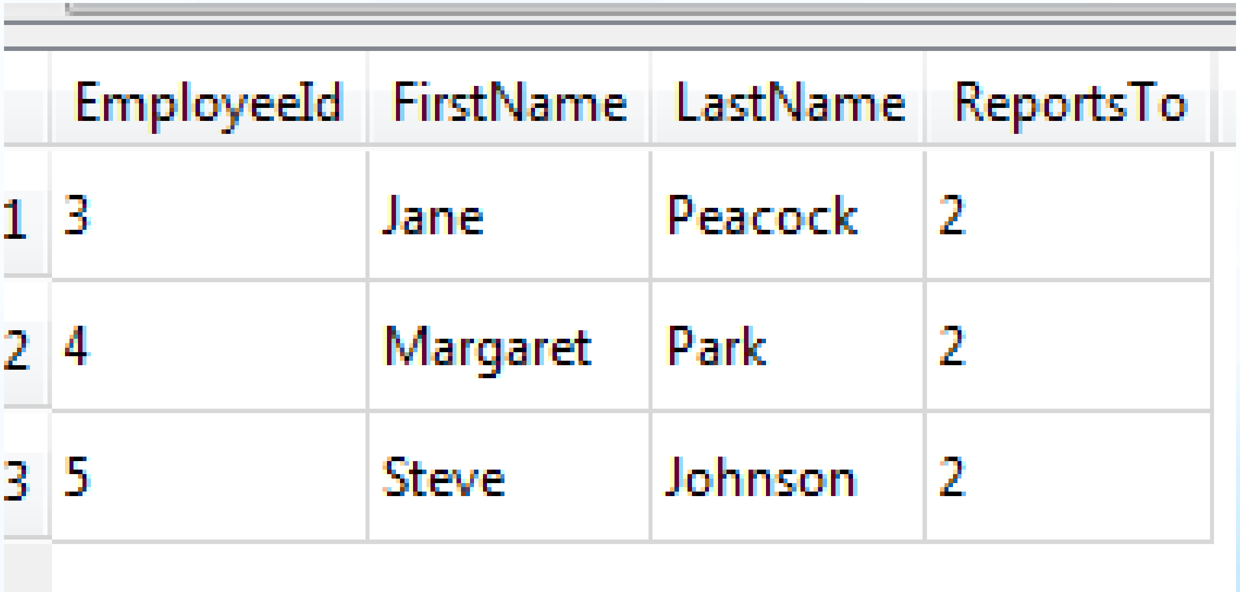
SQL-select

- Ascending and descending order
- select EmployeeId, FirstName, Lastname, Title, reportsto
- from employees
- Order by firstname DESC

EmployeeId	FirstName	LastName	Title	ReportsTo
5	Steve	Johnson	Sales Support Agent	2
7	Robert	King	IT Staff	6
2	Nancy	Edwards	Sales Manager	1
6	Michael	Mitchell	IT Manager	1
4	Margaret	Park	Sales Support Agent	2
8	Laura	Callahan	IT Staff	6
3	Jane	Peacock	Sales Support Agent	2
1	Andrew	Adams	General Manager	NULL

SQL-select

- select EmployeeId, firstname, lastname, reportsto
- from employees
- where reportsto = 2



	EmployeeId	FirstName	LastName	ReportsTo
1	3	Jane	Peacock	2
2	4	Margaret	Park	2
3	5	Steve	Johnson	2

SQL-select

- Where clause operations

operator	Description
=	Equality
<>	Non-equality
!=	Non-equality
<	Less than
<=	Less than or equal
!<	Not less than
>	Greater than
>=	Greater than or equal
!>	Not greater than
Between	Between two specified values
Is NULL	Is a Null value

SQL-select, “AND” & “OR”

- For the following table

Lastname	Firstname	Title	Reportsto	country
Adams	Andrew	General Manager		Iraq
Edwards	Nancy	Sales Manager	1	Iraq
Peacock	Jane	Sales Support Agent	2	Iraq
Park	Margaret	Sales Support Agent	2	Iraq
Johnson	Steve	IT Manager	2	Canada
Mitchell	Michael	IT Manager	1	Canada
King	Robert	IT Staff	6	Canada
Callahan	Laura	IT Staff	6	Canada

There are two different output when I use AND or OR Instructions

SQL-select, “AND” & “OR”

- Using “OR”
  - select lastname, firstname, title, reportsto, country
  - from employees
  - where reportsto=2 or title='IT Manager'

Lastname	Firstname	Title	Reportsto	country
Peacock	Jane	Sales Support Agent	2	Iraq
Park	Margaret	Sales Support Agent	2	Iraq
Johnson	Steve	IT Manager	2	Canada
Mitchell	Michael	IT Manager	1	Canada

SQL-select, “AND” & “OR”

- Using “AND”
  - select lastname, firstname, title, reportsto, country
  - from employees
  - where reportsto=2 and title='IT Manager'

Lastname	Firstname	Title	Reportsto	country
Johnson	Steve	IT Manager	2	Canada

SQL-select, “AND” & “OR”

- Using “AND” & “OR”
  - select lastname, firstname, title, reportsto, country
  - from employees
  - where reportsto=2 or title='IT Manager' and country='iraq'

Lastname	Firstname	Title	Reportsto	country
Peacock	Jane	Sales Support Agent	2	Iraq
Park	Margaret	Sales Support Agent	2	Iraq
Johnson	Steve	IT Manager	2	Canada

Using IN operator

- The “in” operator is used to specify a range of conditions, any of which can be matched.
  - Select lastname, firstname, title, reportsto, country
  - from employees
  - where title IN ('IT Manager' , 'IT Staff ‘)

Lastname	Firstname	Title	Reportsto	country
Johnson	Steve	IT Manager	2	Canada
Mitchell	Michael	IT Manager	1	Canada
King	Robert	IT Staff	6	Canada
Callahan	Laura	IT Staff	6	Canada

Using NOT operator

- The WHERE clause’s NOT operator has one function and one function only NOT negates whatever condition come next
  - select lastname, firstname, title, reportsto, country
  - from employees
  - where NOT Country = 'canada'

Lastname	Firstname	Title	Reportsto	country
Adams	Andrew	General Manager		Iraq
Edwards	Nancy	Sales Manager	1	Iraq
Peacock	Jane	Sales Support Agent	2	Iraq
Park	Margaret	Sales Support Agent	2	Iraq

SQL-LIKE operator

- All previous operator was against known values. It matching one or more value, greater than, less than known values.
- But filtering data by that way does not always work.
- Specially when we want to search for a text inside a word

SQL-LIKE “Percentage Sign %”

- It is most widely Wildcard used within a search.
  - Select lastname, firstname, title, reportsto, country
  - From employees
  - where Title LIKE 'Sales%‘

- For this example is used to find all Titles that start with Sales.
- The % tells the DBMS to accept any characters after the word Sales. Regardless of how many characters there are.

Lastname	Firstname	Title	Reportsto	country
Edwards	Nancy	Sales Manager	1	Iraq
Peacock	Jane	Sales Support Agent	2	Iraq
Park	Margaret	Sales Support Agent	2	Iraq

SQL-LIKE “Percentage Sign %”

- select lastname, firstname, title, reportsto, country
- from employees
- where Title LIKE '%Manager'

Lastname	Firstname	Title	Reportsto	country
Adams	Andrew	General Manager		Iraq
Edwards	Nancy	Sales Manager	1	Iraq
Johnson	Steve	IT Manager	2	Canada
Mitchell	Michael	IT Manager	1	Canada

SQL-LIKE “Percentage Sign %”

- select lastname, firstname, title, reportsto, country
- from employees
- where Title LIKE '%al%'

Lastname	Firstname	Title	Reportsto	country
Adams	Andrew	General Manager		Iraq
Edwards	Nancy	Sales Manager	1	Iraq
Peacock	Jane	Sales Support Agent	2	Iraq
Park	Margaret	Sales Support Agent	2	Iraq

SQL-LIKE Wildcard

- The underscore is used jest like “%” except that “_” matches just a single character.
  - Select lastname, title, reportsto, country, phone
  - From Employees
  - where Title LIKE '_ _ _Manager'
- The brackets “[ ]” is used to specify a set of characters, any which must match a character in the specified position.

Text Manipulation Function

- Select lastname, Upper(title), reportsto, country, phone
- From Employees

Lastname	Title	Reportsto	country	phone
Adams	GENERAL MANAGER		Iraq	7804289482
Edwards	SALES MANAGER	1	Iraq	4032623443
Peacock	SALES SUPPORT AGENT	2	Iraq	4032623443
Park	SALES SUPPORT AGENT	2	Iraq	4032634423
Johnson	IT MANAGER	2	Canada	7808369987
Mitchell	IT MANAGER	1	Canada	4032469887
King	IT STAFF	6	Canada	4034569986
Callahan	IT STAFF	6	Canada	4034673351

- Select lastname, Upper(title), reportsto, country as CITY, phone
- From Emp

Lastname	Title	Reportsto	CITY	phone
Adams	GENERAL MANAGER		Iraq	7804289482
Edwards	SALES MANAGER	1	Iraq	4032623443
Peacock	SALES SUPPORT AGENT	2	Iraq	4032623443
Park	SALES SUPPORT AGENT	2	Iraq	4032634423
Johnson	IT MANAGER	2	Canada	7808369987
Mitchell	IT MANAGER	1	Canada	4032469887
King	IT STAFF	6	Canada	4034569986
Callahan	IT STAFF	6	Canada	4034673351

Text Manipulation Function

- Commonly used Text-Manipulation functions

Function	Description
LEFT ( )	Returns characters from left of string
Length ( ) or Len ( )	Returns the length of a string
LOWER ( )	Converts the string to lowercase
LTRIM ( )	Trims white space from right of string
RIGHT ( )	Returns characters from right of string
RTRIM ( )	Trims white space from right of string
UPPER ( )	Converts string to uppercase

Using Aggregate functions

- Select lastname, title, BirthDate, MIN(Salary) as min_salary
- From Employees
  - MIN ( ) it return the lowest value in a specified column
  - MAX ( ) it return the largest value in a specified column

Lastname	Title	Salary	CITY	phone
Adams	GENERAL MANAGER	3000	Iraq	7804289482
Edwards	SALES MANAGER	2000	Iraq	4032623443
Peacock	SALES SUPPORT AGENT	1000	Iraq	4032623443
Park	SALES SUPPORT AGENT	1000	Iraq	4032634423
Johnson	IT MANAGER	2000	Canada	7808369987
Mitchell	IT MANAGER	2000	Canada	4032469887
King	IT STAFF	1500	Canada	4034569986
Callahan	IT STAFF	1500	Canada	4034673351

Using Aggregate functions

- SUM ( ) it return the sum total of the values in a specified column
  - Select sum(Salary) as total_salary
  - From Employees

  - select sum(itemprice * quantity) as total_sale
  - from Invoices

Invoice ID	Customer name	Unit Price	Quantity
1001	Global	1.99	2
1002	Global	4	5
1003	Airport	10	3
1004	University	12	2
1005	University	6	1
1006	Airport	11	5
1007	IT Center	2	10

- Result

Total_sale
158.98

Using Aggregate functions

- select avg(itemprice) as average_price
- from Invoices

Average_Price
6.712

- select count(invoiceid) as no_of_invoices,
- min(Itemprice) as minimum_price,
- max(itemprice) as maximum_price,
- avg(itemprice) as average_price
- from Invoices

No_of_invoices	Minimum_price	Maximum_price	Average_price
7	1.99	12	6.72

Joining Tables

- As just explained, breaking data into multiple tables enables more efficient storage, easer manipulation, and grater scalability.

- select *
- from invoices, invoice_items
- where invoices.InvoiceId = invoice_items.InvoiceId