Intro To Database

(Database Fundamental using MySQL)

Mohamed ELshafei

Database Normalization

- Normalization: The process of structuring data to minimize duplication and inconsistencies.
- The process usually involves breaking down a single Table into two or more tables and defining relationships between those tables.
- Normalization is usually done in stages, with each stage applying some rules to the types of information which can be stored in a table.

Normalization

- Normalization is a bottom-up Analysis
- Normalization is used to reduce Null Values
- Normalization is used to improve performance

Well-Structured Relations

Goal is to avoid anomalies

- **Insertion Anomaly** – adding new rows forces user to create duplicate data
- **Deletion Anomaly** – deleting rows may cause a loss of data that would be needed for other future rows
- **Modification Anomaly** – changing data in a row forces changes to other rows because of duplication

Example

<u>SID</u>	Sname	Bdate	City	ZipCode	<u>Subject</u>	Grade	Teacher
1	Ahmed	1/1/1980	Cairo	1010	DB	A	Hany
1	Ahmed	1/1/1980	Cairo	1010	Math	B	Eman
1	Ahmed	1/1/1980	Cairo	1010	WinXP	A	khalid
2	Ali	1/1/1983	Alex	1111	DB	B	Hany
2	Ali	1/1/1983	Alex	1111	SWE	B	Heba
3	Mohamed	1/1/1990	Mansoura	1210	NC	C	Mona

Functional dependency

a constraint between two attributes (columns) or two sets of columns

- A => B if "for every valid instance of A, that value of A uniquely determines the value of B"
- or …A => B if "existing of B depending on a value of A"

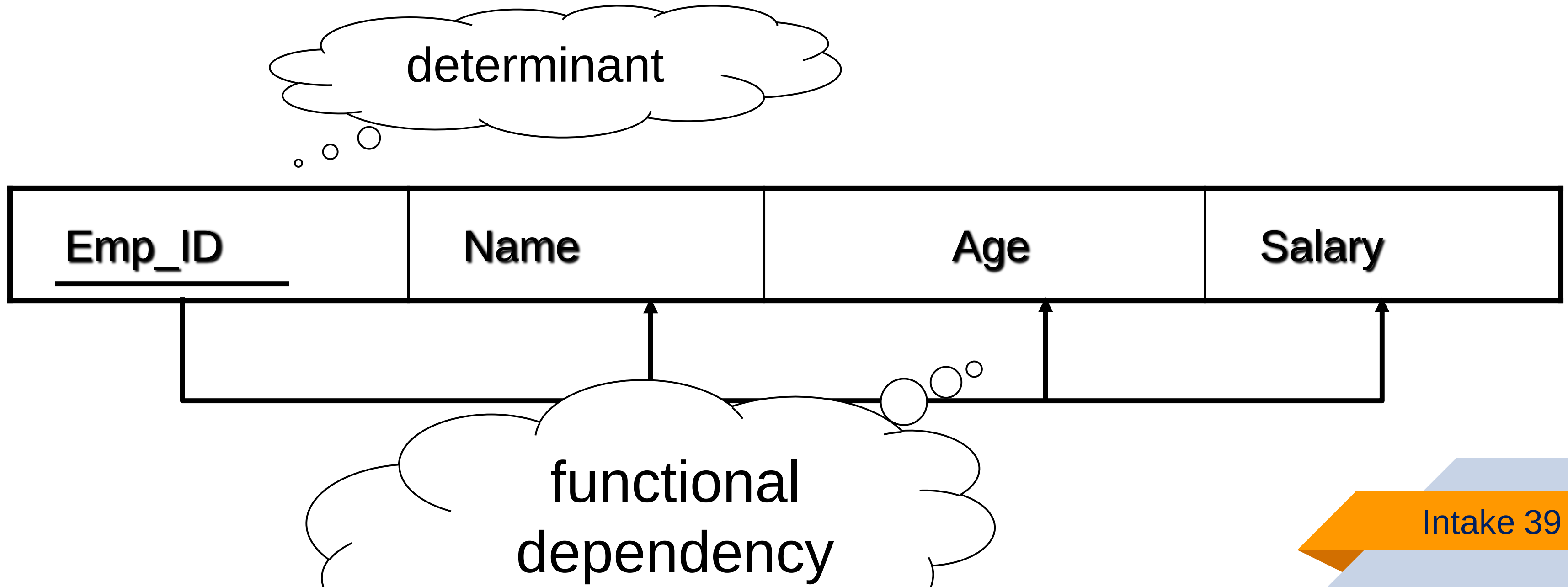
Examples

- Social security number determines employee name SSN -> ENAME
- project number determines project name and location

  PNUMBER -> {PNAME, PLOCATION}

keys and dependencies

EMPLOYEE1 (Emp_ID, Name, Age, Salary)



Types of functional dependency

- **Full Functional Dependency**

Attribute is fully Functional Dependency on a PK if its value is determined by the whole PK

- **Partial Functional Dependency**

Attribute if has a Partially Functional Dependency on a PK if its value is determined by part of the PK(Composite Key)

- **Transitive Functional Dependency**

Attribute is Transitively Functional Dependency on a table if its value is determined by anther non-key attribute which it self determined by PK

Steps in normalization

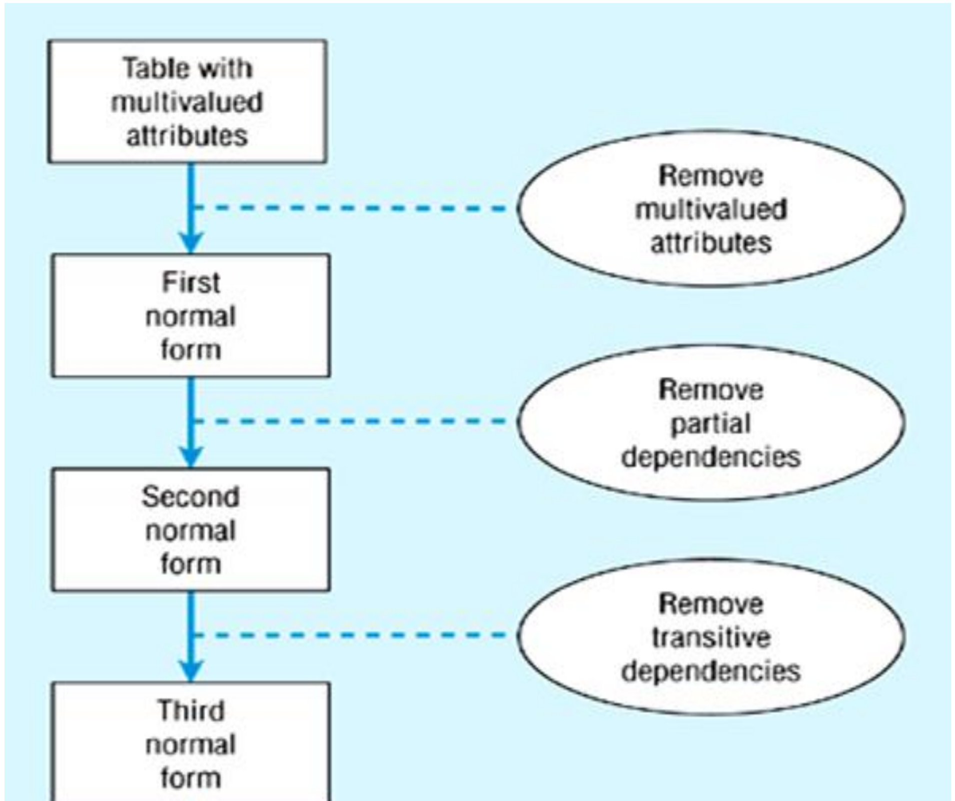


Table with multivalued attributes
Remove multivalued attributes
First normal form
Remove partial dependencies
Second normal form
Remove transitive dependencies
Third normal form

1NF

- relation is in first normal form if it contains no multivalued or composite attributes
- remove repeating groups to a new table as already demonstrated, “carrying” the PK as a FK
- All columns (fields) must be atomic
  - Means : no repeating items in columns

1NF

SID	SName	Birthdate	City	city Code	Subject	Grade	Teacher
1	Ahmed	1/1/1980	Cairo	1010	DB	A	Hany
1	Ahmed	1/1/1980	Cairo	1010	Math	B	Eman
1	Ahmed	1/1/1980	Cairo	1010	WinXP	A	khalid
2	Ali	1/1/1983	Alex	1111	DB	B	Hany
2	Ali	1/1/1983	Alex	1111	SWE	B	Heba
3	Mohamed	1/1/1990	Cairo	1010	NC	C	Mona

SID	SName	Birthdate	City	city Code	Subject	Grade	Teacher
1	Ahmed	1/1/1980	Cairo	1010	DB	A	Hany
					Math	B	Eman
					WinXP	A	khalid
2	Ali	1/1/1983	Alex	1111	DB	B	Hany
					SWE	B	Heba
3	Mohamed	1/1/1990	Cairo	1010	NC	C	Mona

Repeating Groups
Or multivalued

1NF

Student(<u>SID</u>, Sname, Birthdate, City, city code)

<u>SID</u>	SName	Birthdate	City	city Code
1	Ahmed	1/1/1980	Cairo	1010
2	Ali	1/1/1983	Alex	1111
3	Mohamed	1/1/1990	Cairo	1010

Stud_Subject (<u>SID, Subject</u>, Grade, Teacher)

<u>SID</u>	<u>Subject</u>	Grade	Teacher
1	DB	A	Hany
1	Math	B	Eman
1	WinXP	A	khalid
2	DB	B	Hany
2	SWE	B	Heba
3	NC	C	Mona

2NF

- a relation is in **second normal form** if it is in first normal form AND every nonkey attribute is fully functionally dependant on the primary key
- remove partial functional dependencies, so no nonkey attribute depends on just part of the key

2NF

Student(SID, Sname, Birthdate, City, city Code)

SID	SName	Birthdate	City	city Code
1	Ahmed	1/1/1980	Cairo	1010
2	Ali	1/1/1983	Alex	1111
3	Mohamed	1/1/1990	Cairo	1010

2NF
Because there is no Composite PK

Stud_Subject (SID, Subject, Grade, Teacher)

SID	Subject	Grade	Teacher
1	DB	A	Hany
1	Math	B	Eman
1	WinXP	A	khalid
2	DB	B	Hany
2	SWE	B	Heba
3	NC	C	Mona

SID, Subject → Grade……FFD

Subject → Teacher……PFD

2NF

Student(SID, Sname, Birthdate, City, city Code)

SID	SName	Birthdate	City	City Code
1	Ahmed	1/1/1980	Cairo	1010
2	Ali	1/1/1983	Alex	1111
3	Mohamed	1/1/1990	Mansoura	1210

Stud_Subject (SID, Subject, Grade)

SID	Subject	Grade
1	DB	A
1	Math	B
1	WinXP	A
2	DB	B
2	SWE	B
3	NC	C

Subject (Subject,Teacher)

Subject	Teacher
DB	Hany
Math	Eman
WinXP	khalid
SWE	Heba
NC	Mona

3NF

- 2NF PLUS no transitive dependencies (one attribute functionally determines a second, which functionally determines a third)

3NF

Student(SID, Sname, Birthdate, City, city Code)

SID	SName	Birthdate	City	city Code
1	Ahmed	1/1/1980	Cairo	1010
2	Ali	1/1/1983	Alex	1111
3	Mohamed	1/1/1990	Cairo	1010

city Code ->City …….TFD

Stud_Subject (SID, Subject, Grade)

SID	Subject	Grade
1	DB	A
1	Math	B
1	WinXP	A
2	DB	B
2	SWE	B
3	NC	C

Subject (Subject,Teacher)

Subject	Teacher
DB	Hany
Math	Eman
WinXP	khalid
SWE	Heba
NC	Mona

3NF

Because there is no Transtive Functional Dependency

3NF

Student(SID, Sname, Birthdate,city code)

SID	SName	Birthdate	cityCode
1	Ahmed	1/1/1980	1010
2	Ali	1/1/1983	1111
3	Mohamed	1/1/1990	1010

Stud_City(City, city Code)

City	city Code
Cairo	1010
Alex	1111

Stud_Subject (SID, Subject, Grade)

SID	Subject	Grade
1	DB	A
1	Math	B
1	WinXP	A
2	DB	B
2	SWE	B
3	NC	C

Subject (Subject,Teacher)

Subject	Teacher
DB	Hany
Math	Eman
WinXP	khalid
DB	Hany
SWE	Heba
NC	Mona

ITI Case study

ITI Students Sheet

Platform Name : SWE **Platform Description:** Software Engineering

Graduate Manager: Dr. Baha

Appno	Name	F-code	Faculty	Address	Telno	Grade	Att. Hrs	Sdate
123	Ahmed	SC-phy	Science	Haram	3386842	A	600	14 Sep
124	Mona	Eng-cs	Engineering	Dokki	3389745, 3389744, 5123445	B	591	15 Sep
127	Ali	Com-ac	Commerce	Nasr City	2241593, 2222345	A	550	21 Sep
223	Karim	Med-bio	Medicine	Sheraton	2286845	C	600	14 Sep

1NF

- **Platform** :pfname , pfdesc , pfManager
- **Students**: pfname, appno, name , faculty , F-Code, address, grade, attd , start-date
- **Std-Tel**: appno, telno

2NF

- **Students**: appno, name , faculty , FCode, address
- **Students-pf**: pfname,appno, grade, attd , start_date

Unchanged Tables

- **Platform** :pfname , pfdesc , pfManager
- **Std-Tel**: appno, telno

3NF

- **Students**: appno, name , FCode, address
- **Fac-majors**:faculty , FCode

Unchanged Tables

- **Platform** :pfname , pfdesc , pfManager
- **Std-Tel**: appno, telno
- **Students-pf**: pfname,appno, grade, attd , start-date

DML -Data Manipulation Language

- Insert.
- Update.
- Delete.

INSERT Command

Person table

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo

✓ INSERT INTO "table_name” VALUES ("value1", "value2", ...)

- **Insert a New Row:**

  INSERT INTO Person VALUES (‘Saleh’, ‘Ahmed', ‘Moharam bak', ‘Alex.')

Person table

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Saleh	Ahmed	Moharam bak.	Alex.

INSERT Command (cont.)

Person table

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo

- **Insert a New Row:**

INSERT INTO Person (LastName, City) VALUES (‘Hassan', ‘Assuit')

Person table

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Hassan			Assuit.

Update Command

✓ UPDATE "table_name"
SET "column_1" = {new value}
[WHERE {condition}]

<u>Example (1)</u>

```
UPDATE Person
SET City= ‘Assiut’
```

All records will be updated

<u>Example (2)</u>

```
UPDATE Person
SET City= ‘Assiut’

Where FirstName = ‘Ahmed’
```

Only records with first name ‘Ahmed’ will be updated

Update Command (cont.)

✓ Update several Columns in a Row:

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Saleh	Ahmed	Moharam bak.	Alex.

```
UPDATE Person
SET      Address = ‘241 El-haram ',  City = ‘Giza'
WHERE   LastName = ‘El-Sayed'
```

LastName	FirstName	Address	City
El-Sayed	Mohamed	241 El-haram	Giza
Saleh	Ahmed	Moharam bak.	Alex.

Delete Command

✓ DELETE FROM "table_name"
[WHERE {condition}]

Example (1)

```
DELETE FROM Person
```

All records will be deleted

Example (2)

```
DELETE FROM Person
Where FirstName = ‘Ahmed’
```

Only records with first name ‘Ahmed’ will be deleted

DQL

```
Select <attribute list >
From  < table list>
[ Where <condition> ]
```

✓ ```
select *
from department;
```

✓ ```
select emp_id, emp_name, dept_id
from   employee;
```

✓ ```
select distinct dept_id
from employee;
```
```

SELECT with Condition

```
Select  dept_id, dept_name
from    department
where   location = 'Cairo';
```

Comparison Conditions

- = Equal.
- > greater than.
- >= greater than or equal.
- < less than.
- <= less than or equal.
- <>not equal.

```
Select   last_name, salary
from     employee
where    salary >1000
```

Logical Conditions

- AND.

```
Select   last_name, salary
from     employee
where  city = ‘Assiut’ and salary > 1000;
```

- OR.

```
Select   last_name, salary
from     employee
where  city = ‘Assiut’ OR salary > 1000;
```

- NOT.

```
Select   emp_id, last_name, salary, manager_id
From     employee
where   manager_id NOT IN (100, 101, 200);
```

Other Comparison Conditions

- Between …… AND ….. (between two values - Inclusive).

```
Select   last_name, salary
from     employee
where  salary between 1000 and 3000;
```

- IN (set) (Match any of a list of values)

```
Select    emp_id, last_name, salary, manager_id
From      employee
where     manager_id IN (100, 101, 200);
```

- Like (Match a character Pattern)

```
Select   first_name
from     employee
where    first_name Like ‘s%’;
```

Arithmetic Expressions

```
Select   last_name, salary, salary + 300
from     employee;
```

- Order of precedence: * , / , +, -
- You can enforce priority by adding parentheses.

```
Select    last_name, salary, 10 * (salary + 300)
from      employee;
```

Order by Clause

- It is used to sort results either in ascending or descending order.

```
✓ Select     fname, dept_id, hire_date
  From       employee
  Order by   hire_date  [ ASC ];

✓ Select     fname, dept_id, hire_date
  From       employee
  Order by   hire_date  DESC;

✓ Select     fname, dept_id, salary
  From       employee
  Order by   dept_id, Salary  DESC;
```

THANKS!

Any questions?