MONASH University

MONASH
INFORMATION
TECHNOLOGY

# Normalisation



GROUP OF EIGHT AUSTRALIA

# Data Normalisation

- Relations should be normalised in order to avoid anomalies which may occur when inserting, updating and deleting data.
- Normalisation is a **systematic series of steps** for progressively refining the data model.
- A formal approach to analysing relations based on their primary key (or candidate keys) and functional dependencies.
- Used:
  - as a design technique "bottom up design", and
  - as a way of validating structures produced via "top down design" (ER model converted to a logical model - see next session)

# Sample Data

| PROJ_NUM | PROJ_NAME | EMP_NUM | EMP_NAME | JOB_CLASS | CHG_HOUR | HOURS |
|---|---|---|---|---|---|---|
| 15 | Evergreen | 103 | June E. Arbough | Elect. Engineer | 84.50 | 23.80 |
| 15 | Evergreen | 101 | John G. News | Database Designer | 105.00 | 19.40 |
| 15 | Evergreen | 105 | Alice K. Johnson * | Database Designer | 105.00 | 35.70 |
| 15 | Evergreen | 106 | William Smithfield | Programmer | 35.75 | 12.60 |
| 15 | Evergreen | 102 | David H. Senior | Systems Analyst | 96.75 | 23.80 |
| 18 | Amber Wave | 114 | Annelise Jones | Applications Designer | 48.10 | 24.60 |
| 18 | Amber Wave | 118 | James J. Frommer | General Support | 18.36 | 45.30 |
| 18 | Amber Wave | 104 | Anne K. Ramoras * | Systems Analyst | 96.75 | 32.40 |
| 18 | Amber Wave | 112 | Darlene M. Smithson | DSS Analyst | 45.95 | 44.00 |
| 22 | Rolling Tide | 105 | Alice K. Johnson | Database Designer | 105.00 | 64.70 |
| 22 | Rolling Tide | 104 | Anne K. Ramoras | Systems Analyst | 96.75 | 48.40 |
| 22 | Rolling Tide | 113 | Delbert K. Joenbrood * | Applications Designer | 48.10 | 23.60 |
| 22 | Rolling Tide | 111 | Geoffrey B. Wabash | Clerical Support | 26.87 | 22.00 |
| 22 | Rolling Tide | 106 | William Smithfield | Programmer | 35.75 | 12.80 |
| 25 | Starflight | 107 | Maria D. Alonzo | Programmer | 35.75 | 24.60 |
| 25 | Starflight | 115 | Travis B. Bawangi | Systems Analyst | 96.75 | 45.80 |
| 25 | Starflight | 101 | John G. News * | Database Designer | 105.00 | 56.30 |
| 25 | Starflight | 114 | Annelise Jones | Applications Designer | 48.10 | 33.10 |
| 25 | Starflight | 108 | Ralph B. Washington | Systems Analyst | 96.75 | 23.60 |
| 25 | Starflight | 118 | James J. Frommer | General Support | 18.36 | 30.50 |
| 25 | Starflight | 112 | Darlene M. Smithson | DSS Analyst | 45.95 | 41.40 |



* against EMP_NAME indicates the project leader

# Problems with sample data

- JOB_CLASS invites **entry errors** eg. Elec. Eng. vs Elect. Engineer vs E.E.
- Table has **redundant data**
  - Details of a charge per hour are repeated for every occurrence of job class
  - Every time an employee is assigned to a project emp name repeated
- Relations that contain redundant information may potentially suffer from several update anomalies
  - Types of update anomalies include:
    - **Insert Anomaly**
      - Insert a new employee only if they are assigned to a project
    - **Delete Anomaly**
      - Delete the only employee assigned to a project?
      - Delete the only employee of a particular job class?
    - **Modification (or update) Anomaly**
      - Update a job class hourly rate - need to update multiple rows

# The Normalisation Process Goals

- Creating valid relations, i.e. each relation meets the properties of the relational model. In particular:
  - Entity integrity
  - Referential integrity
  - No many-to-many relationship
  - Each cell contains a single value (is atomic).
- In practical terms when implemented in an RDBMS:
  - Each table represents a single subject
  - No data item will be unnecessarily stored in more than one table.
  - The relationship between tables can be established (pair of PK and FK is identified).
  - Each table is void of insert, update and delete anomalies.

# Representing a form as a relation

- This process follows a **standard** approach:
  - arrive at a name for the form which indicates what it represents (its subject)
  - determine if any attribute is multivalued (repeating) **for a given entity instance of the forms subject**
    - if an attribute (or set of attributes) appears multiple times then the group of related attributes need to be shown enclosed in brackets to indicate there are multiple sets of these values for each instance
- Looking at our SAMPLE DATA
  - Name: EMPLOYEE_PROJECT_ASSIGNMENT
    - simplify name to ASSIGNMENT for lecture
  - ASSIGNMENT (proj_num, emp_num, emp_name, job_class, chg_hour, assign_hours)
  - i.e. the form consists of repeating rows (instances) of assignment data

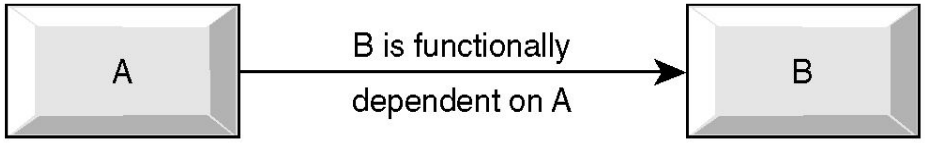
# Representing a form as a relation

| CUSTOMER ORDER | | | |
|---|---|---|---|
| **Order Number:** | 61384 | | **Order Date:** 12/3/2020 |
| **Customer Number:** | 1273 | | |
| **Customer Name:** | Computer Training Centre | | |
| **Customer Address:** | 123 Excellent St<br>Monash, Vic 3000 | | |
| **PRODUCT NUMBER** | **DESCRIPTION** | **QTY ORDERED** | **LINE PRICE** |
| M128 | Bookcase | 4 | 800 |
| B381 | TV Cabinet | 2 | 600 |
| R210 | Round Table | 3 | 1500 |



**ORDER** ( orderno, orderdate, custnumb, custname, custaddress (prodno, proddesc, qtyordered, lineprice))

# Functional Dependency Revisited




- An attribute B is FUNCTIONALLY DEPENDENT on another attribute A, if a value of A determines a single value of B at any one time.
  - A ➔ B
  - PRODNO ➔ PRODDESC
  - CUSTNUMB ➔ CUSTNAME
  - ORDERNO ➔ ORDERDATE
    - ORDERNO - independent variable, also known as the DETERMINANT
    - ORDERDATE - dependent variable
- TOTAL DEPENDENCY
  - attribute A determines B AND attribute B determines A
    - EMPLOYEE-NUMBER ➔ TAX-FILE-NUMBER
    - TAX-FILE-NUMBER ➔ EMPLOYEE-NUMBER

# Functional Dependency

- For a ***composite*** PRIMARY KEY, it is possible to have FULL or PARTIAL dependency.
- FULL DEPENDENCY
  - occurs when an attribute is always dependent on all attributes in the composite PK
  - ORDERNO, PRODNO → QTYORDERED
- Lack of full dependency for multiple attribute key = **PARTIAL DEPENDENCY**
  - ORDERNO, PRODNO
    → PRODDESC, QTYORDERED
  - here although qtyordered is ***fully dependent*** on orderno and prodno, *only* prodno is required to determine proddesc
  - proddesc is said to be ***partially dependent*** on orderno and prodno

# Functional Dependency

- **TRANSITIVE DEPENDENCY**
  - occurs when Y depends on X, and Z depends on Y - thus Z also depends on X ie. X ➔ Y ➔ Z
  - **and** Y is not a candidate key (or part of a candidate key)
  - ORDERNO ➔ CUSTNUMB ➔ CUSTNAME
- Dependencies are depicted with the help of a **Dependency Diagram**.
- Normalisation converts a relation into relations of progressively smaller number of attributes and tuples until an optimum level of decomposition is reached - little or no data redundancy exists.
- The output from normalisation is a set of relations that meet all conditions set in the relational model principles.

# Unormalised Form (UNF)

- The UNF representation of a relation is the representation which you have mapped from your inspection of the form
  - it is a **single** named representation (name is not pluralised)
  - no PK etc have as yet been identified
- **ASSIGNMENT** (proj_num, emp_num, emp_name, job_class, chg_hour, assign_hours)
- **ORDER** (orderno, orderdate, custnumb, custname, custaddress (prodno, proddesc, qtyordered, lineprice))



Can ASSIGNMENT and/or ORDER be called a relation? If not, why not?

# First Normal Form

- FIRST NORMAL FORM (part of formal definition of a relation)
  - A RELATION IS IN FIRST NORMAL FORM (1NF) IF:
    - a unique primary key has been identified for each tuple/row.
    - it is a valid relation
      - Entity integrity (no part of PK is null)
      - Single value for each cell ie. no repeating group (multivalued attribute).
    - all attributes are functionally dependent on all or part of the primary key

# UNF to 1NF

- Move from UNF to 1NF by:
  1. identify a unique identifier for the repeating group.
  2. *remove any repeating group* along with the PK of the main relation.
  3. The PK of the new relation resulting from the removal of repeating group will *normally* have a composite PK made up of the PK of the main relation and the unique identifier chosen in 1. above, but this ***must be checked***.

**Q1. Given the CUSTOMER ORDER UNF as:**

**ORDER ( orderno, orderdate, custnumb, custname, custaddress (prodno, proddesc, qtyordered, lineprice)).**

**What would be the 1NF of this UNF relation?**

A. Two relations
- ORDER (orderno, orderdate, custnumb, custname, custaddress)
- ORDER_PROD (orderno, prodno, proddesc, qtyordered, lineprice )

B. Three relations
- ORDER (orderno, orderdate, custnumb)
- CUSTOMER (custnumb, custname, custaddress)
- ORDER_PROD (orderno, prodno, proddesc, qtyordered, lineprice )

C. CUST_ORDER (orderno, orderdate, custnumb, custname, custaddress, prodno, proddesc, qtyordered, lineprice)

D. PROD_ORDER (orderno, prodno, proddesc, qtyordered, lineprice, orderdate, custnumb, custname, custaddress)

## Q2. Given the ASSIGNMENT UNF as:

**ASSIGNMENT (proj_num, emp_num, proj_name, emp_name, job_class, chg_hour, assign_hours).**

**What would be the 1NF of this UNF relation?**



A. Two relations
   - PROJECT (proj_num, proj_name) and
   - ASSIGNMENT (proj_num, emp_num, emp_name, job_class, chg_hour, assign_hours)

B. ASSIGNMENT (proj_num, emp_num, proj_name, emp_name, job_class, chg_hour, assign_hours)

C. PROJECT (proj_num, proj_name)

D. ASSIGNMENT (proj_num, emp_num, job_class, proj_name, emp_name, chg_hour, assign_hours)

# 1NF to 2NF

- A RELATION IS IN 2NF IF -
  - all non key attributes are functionally dependent on the primary key (simple definition)
    - used by the textbook in examples
  - all non key attributes are functionally dependent on **any candidate key** (general definition)
    - see textbook section 6-3, same as *simple* if only one candidate key
    - **Requirement for our unit**

## Q3. Which of the following attributes has a partial dependency in the relation ASSIGNMENT?:

**ASSIGNMENT(proj_num, emp_num, proj_name, emp_name, job_class, chg_hour, assign_hours)**



A. proj_name
B. emp_name
C. job_class
D. chg_hour
E. assign_hours
F. More than one option is correct.

## Q4. Which of the following attributes has a transitive dependency in the relation ASSIGNMENT?:

**ASSIGNMENT(proj_num, emp_num, proj_name, emp_name, job_class, chg_hour, assign_hours)**

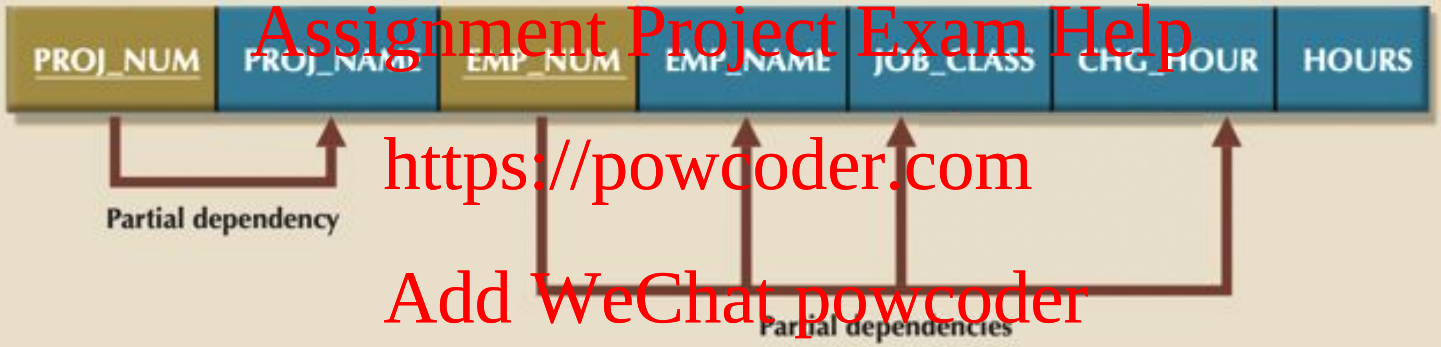


A. proj_name
B. emp_name
C. job_class
D. chg_hour
E. assign_hours
F. More than one option is correct.

# Dependency Diagram ( Drawn based on 1NF)

**Note show only partial dependencies at 1NF**

FIGURE 6.3 FIRST NORMAL FORM (1NF) DEPENDENCY DIAGRAM






1NF (PROJ_NUM, EMP_NUM, PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOURS, HOURS)

PARTIAL DEPENDENCIES:
(PROJ_NUM → PROJ_NAME)
(EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR)

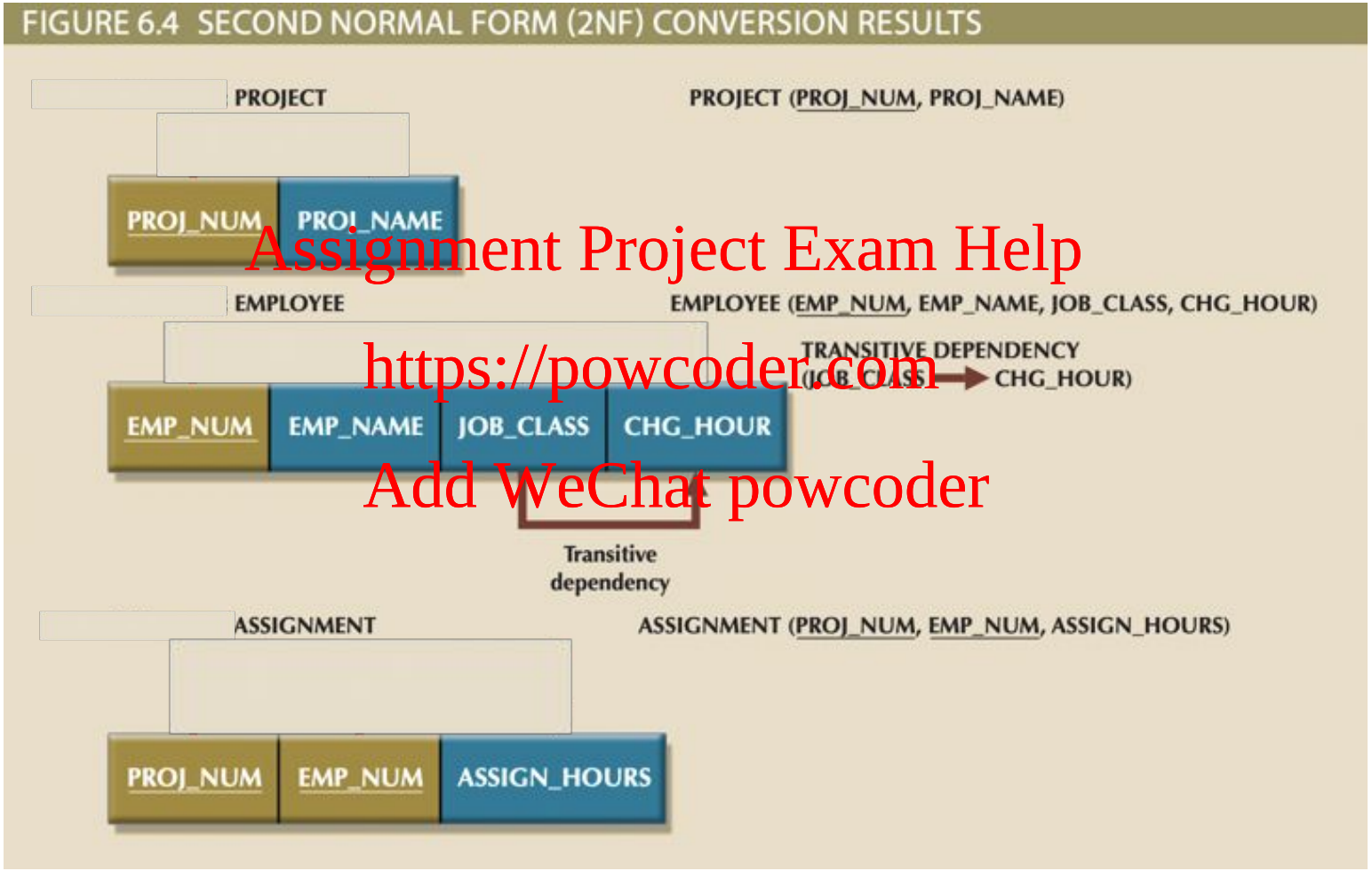
# 1NF to 2NF

- ASSIGNMENT(<u>proj_num</u>, <u>emp_num</u>, proj_name, emp_name, job_class, chg_hour, assign_hours)

- Move from 1NF to 2NF by removing partial dependencies
  - 2NF: ASSIGNMENT (<u>proj_num, emp_num</u>, assign_hours)
  - 2NF: PROJECT (<u>proj_num</u>, proj_name)
  - 2NF: EMPLOYEE (<u>emp_num</u>, emp_name, job_class, chg_hour)

# 2NF Conversion Results

**Note show only transitive dependencies at 2NF**



## Q5. Where is the location of the FK for the relations below?

**EMPLOYEE (emp_num, emp_name, job_class, chg_hour)**
**ASSIGNMENT (proj_num, emp_num, assign_hours)**
**PROJECT (proj_num, proj_name)**



A. EMPLOYEE
B. ASSIGNMENT
C. PROJECT
D. More than one answer is correct

**Q6. What type of relationship is the relationship between:**

**ASSIGNMENT and EMPLOYEE**

**and**

**ASSIGNMENT and PROJECT**



**EMPLOYEE (emp_num, emp_name, job_class, chg_hour)**
**ASSIGNMENT (proj_num, emp_num, assign_hours)**
**PROJECT (proj_num, proj_name)**





A. non-identifying, non-identifying
B. identifying, identifying
C. identifying, non-identifying
D. non-identifying, identifying

# 2NF to 3NF

- A RELATION IS IN 3NF IF -
  - all transitive dependencies have been removed
    - check for ***non key attribute dependent on another non key attribute***
- Move from 2NF to 3NF by removing transitive dependencies

# 2NF to 3NF

- PROJECT and ASSIGN already in 3NF
  - 3NF PROJECT (proj_num, proj_name)
  - 3NF ASSIGNMENT (proj_num, emp_num, assign_hours)
- 2NF EMPLOYEE (emp_num, emp_name, job_class, chg_hour)
  - It has transitive dependency, job_class-> chg_hour.
    - Remove the attributes with transitive dependency into a new relation.
    - The determinant will be an attribute in both the original and new relations (it will become the PK and FK relationship)
    - Assign the determinant to be the PK of the new relation.

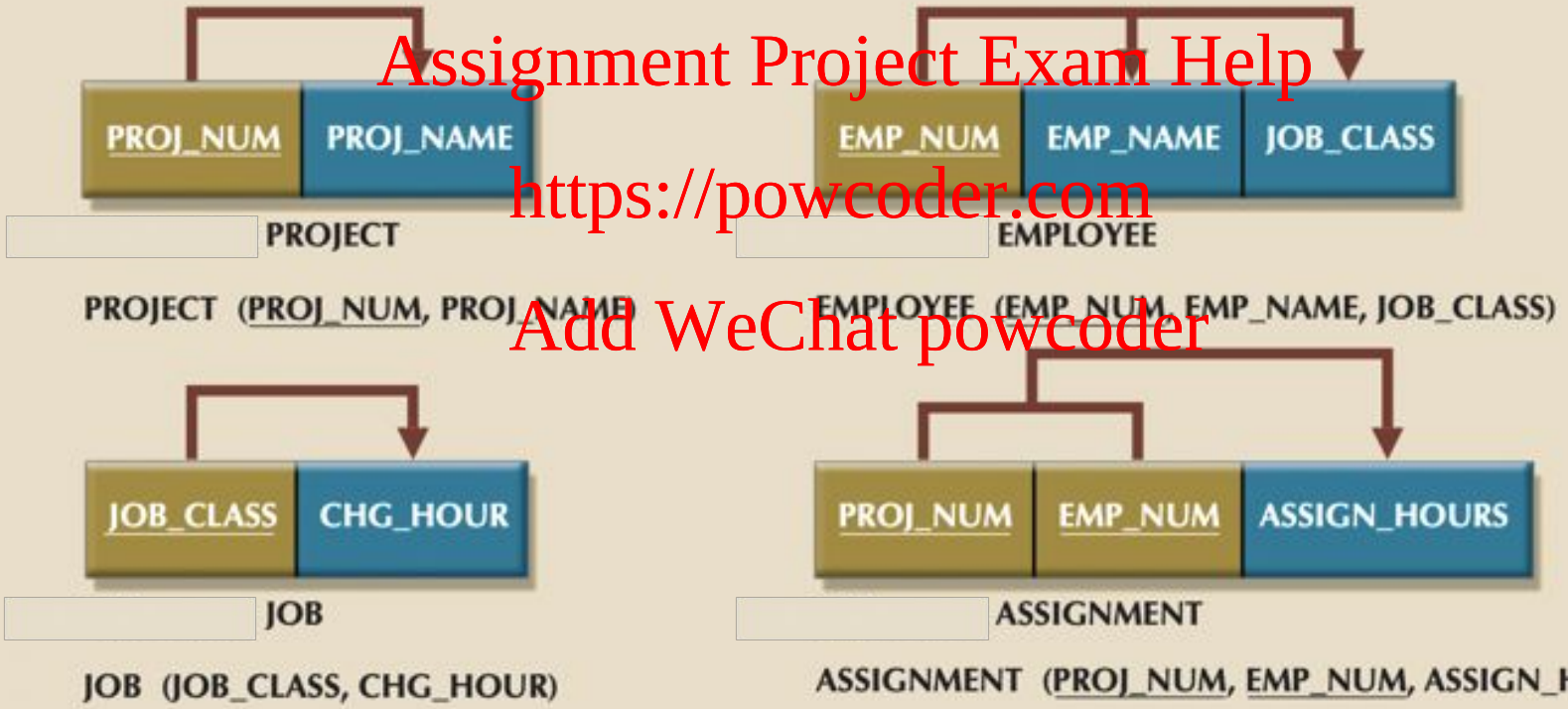
# 2NF to 3NF

- After the removal of transitive dependency in EMPLOYEE, we have:
  - 3NF EMPLOYEE (emp_num, emp_name, job_class)
  - 3NF JOB (job_class, chg_hour)

# Relations in 3NF

**Note show only full dependencies at 3NF**

FIGURE 6.5 THIRD NORMAL FORM (3NF) CONVERSION RESULTS



## Q7. Where is the location of the FK for the relations below?

**EMPLOYEE (emp_num, emp_name, job_class)**
**JOB (job_class, chg_hour)**



A. EMPLOYEE
B. JOB
C. Both EMPLOYEE and JOB

**Q8. What type of relationship is the relationship between the JOB and EMPLOYEE?**

**EMPLOYEE (emp_num, emp_name, job_class)**
**JOB (job_class, chg_hour)**



A. non-identifying
B. identifying
C. Cannot be determined

# Entire Process UNF to 3NF

- UNF
  - ASSIGNMENT (proj_num,emp_num ,proj_name, emp_name, job_class, chg_hour, assign_hours)
- 1NF – remove repeating group
  - ASSIGNMENT (proj_num,emp_num, proj_name, emp_name, job_class, chg_hour, assign_hours) Note: 1NF is only identify PK, no repeating group.
- 2NF – remove partial dependencies
  - ASSIGNMENT (proj_num, emp_num, assign_hours)
  - PROJECT (proj_num, proj_name)
  - EMPLOYEE (emp_num, emp_name, job_class, chg_hour)
- 3NF – remove transitive dependencies
  - ASSIGNMENT (proj_num, emp_num, assign_hours)
  - PROJECT (proj_num, proj_name)
  - EMPLOYEE (emp_num, emp_name, job_class)
  - JOB (job_class, chg_hour)
- *NOTE: dependencies **must be shown at each normal form**, not shown here*

# Customer Order Form

**CUSTOMER ORDER**

**Order Number:** 61384 **Order Date:** 12/3/2020

**Customer Number:** 1273
**Customer Name:** Computer Training Centre
**Customer Address:** 123 Excellent St
Monash, Vic 3000



| PRODUCT NUMBER | DESCRIPTION | QTY ORDERED | LINE PRICE |
|---|---|---|---|
| M128 | Bookcase | 4 | 800 |
| B381 | TV Cabinet | 2 | 600 |
| R210 | Round Table | 3 | 1500 |





**ORDER** ( orderno, orderdate, custnumb, custname, custaddress (prodno, proddesc, qtyordered, lineprice))

# Customer Order Normalisation

**UNF**
**ORDER** ( orderno, orderdate, custnumb, custname, custaddress (prodno, proddesc, qtyordered, lineprice))

**1NF**
**ORDER** ( orderno, orderdate, custnumb, custname, custaddress)
**ORDER_PRODUCT** ( orderno, prodno, proddesc, qtyordered, lineprice)
Partial dependencies:
prodno -> proddesc



**2NF**
**ORDER** ( orderno, orderdate, custnumb, custname, custaddress)
**ORDER_PRODUCT** ( orderno, prodno, qtyordered, lineprice)
**PRODUCT** (prodno, proddesc)
Transitive dependencies:
custnumb -> custname, custaddress

# Customer Order Normalisation continued

**2NF**
**ORDER** ( orderno, orderdate, custnumb, custname, custaddress)
**ORDER_PRODUCT** ( orderno, prodno, qtyordered, lineprice)
**PRODUCT** (prodno, proddesc)
Transitive dependencies:
custnumb -> custname, custaddress



**3NF**
**ORDER** ( orderno, orderdate, custnumb)
**ORDER_PRODUCT** ( orderno, prodno, qtyordered, lineprice)
**PRODUCT** (prodno, proddesc)
**CUSTOMER** (custnumb, custname, custaddress)
Full dependencies:
orderno -> orderdate, custnumb
orderno, prodno -> qtyordered, lineprice
prodno -> proddesc
custnumb -> custname, custaddress

## EMPLOYEE ON-BOARDING FORM

| | | | | |
|---|---|---|---|---|
| **Employee Number** | **1123 (office use only)** | | | |
| **First Name** | Ada | **Last Name** | Lovelace | |
| **DOB** | 1-Jan-1990 | | | |
| **Address** | *Street No* | *Street* | *Suburb* | *Postcode* |
| | 900 | Dandenong Rd | Caulfield East | 3145 |
| **Phone** | 04113344556 (M), 99031000 (OFFICE) | | | |
| **Qualifications** | | | | |
| | **Degree Name** | **Institution** | **Year** | |
| | Bachelor of Computer Science | MIT | 2011 | |
| | Master of Information Technology | Monash | 2013 | |
| **Family Members** | | | | |
| | **No** | **Name** | DOB | |
| | 1 | Albert Einstein | 02-Jan-1992 | |
| | 2 | Grace Hopper | 12-May-1994 | |
| | | | | |
| **SKILL (tick selected)** | | | | |
| | **Skill name** | | | |
| | Java | | | |
| | SQL | | | |
| | SPARK | | | |
| | Python | | | |



*Assume a phone number may be shared between employees*

# Monash Software EMPLOYEE form

- List all attributes found on the form, maintain consistency with previously used attribute names if exist:
  - emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode, phone_type, phone_no, degree_name, degree_institution, degree_year, fmemb_no, fmemb_name, fmemb_dob, skill_name
- Determine if any attribute is multivalued (repeating) for a given entity instance
  - phone_type, phone_no, degree_name, degree_institution, degree_year, fmemb_no, fmemb_name, fmemb_dob, skill_name

# Monash Software EMPLOYEE form continued

- Group multivalued attributes that are related and place in brackets

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode, (phone_type, phone_no), (degree_name, degree_institution, degree_year), (fmemb_no, fmemb_name, fmemb_dob), (skill_name))



- This is our beginning UNF, to proceed to 1NF:
  - PK of main relation EMPLOYEE is emp_no
  - Four repeating groups to remove
    - Remove repeating group (multi valued attribute/s) along with PK of main relation (here emp_no)
  - assume a phone number may be shared between employees

# Monash Software EMPLOYEE form continued

**UNF**

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode, (phone_type, phone_no), (degree_name, degree_institution, degree_year), (fmemb_no, fmemb_name, fmemb_dob), (skill_name))



**1NF**

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode)



EMP_PHONE (emp_no, phone_no, phone_type)



EMP_QUALIFICATION (emp_no, degree_name, degree_institution, degree_year)

FAMILY_MEMBER (emp_no, fmemb_no, fmemb_name, fmemb_dob)

EMPLOYEE_SKILL (emp_no, skill_name)

Partial dependencies:

None present

*Note we are making an assumption that a phone number may be shared between employees*

# Monash Software EMPLOYEE form continued

**2NF**

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode)

EMP_PHONE (emp_no, phone_no, phone_type)

EMP_QUALIFICATION (emp_no, degree_name, degree_institution, degree_year)

FAMILY_MEMBER (emp_no, fmemb_no, fmemb_name, fmemb_dob)

EMPLOYEE_SKILL (emp_no, skill_name)



Transitive dependencies:

None present

# Monash Software EMPLOYEE form continued

**3NF**

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode)

EMP_PHONE (emp_no, phone_no, phone_type)

EMP_QUALIFICATION (emp_no, degree_name, degree_institution, degree_year)

FAMILY_MEMBER (emp_no, fmemb_no, fmemb_name, fmemb_dob)

EMPLOYEE_SKILL (emp_no, skill_name)



Full dependencies:

emp_no -> emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode

emp_no, phone_no -> phone_type

emp_no, degree_name, degree_institution -> degree_year

emp_no, fmemb_no -> fmemb_name, fmemb_dob

# Summary

- Things to remember
  - Represent form as presented, no interpretation, to yield starting point (UNF)
  - Functional dependency
  - Process of removing attributes in relations based on the concept of 1NF, 2NF and 3NF.
    - UNF to 1NF define PK & remove repeating group.
    - 1NF to 2NF remove partial dependency.
    - 2NF to 3NF remove transitive dependency.