Griffith UNIVERSITY

1814ict/2814ict/7003ict/1011ICT: Data Management/ Database Design/ Applied Computing

Revision

Convenor: AProf. Henry Nguyen

School of Information and Communication Technology

Course developed by: Dr Mohammad Awrangjeb; AProf John Wang; Dr Zhe Wang

Entity Relationship Diagram (ERD)

ERD

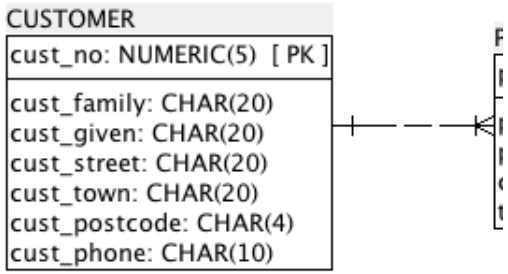
Expressed as a

DBMS Schema:

```
CREATE TABLE CUSTOMER (
    cust_no        INT(5) NOT NULL,
    cust_family    CHAR(20) NOT NULL,
    cust_given     CHAR(20) NOT NULL,
    cust_street    CHAR(20) NOT NULL,
    cust_town      CHAR(20) NOT NULL,
    cust_postcode  CHAR(4) NOT NULL,
    cust_phone     CHAR(10),
CONSTRAINT pk_CUSTOMER PRIMARY KEY
(cust_no)); …etc
```

Or,

Visually using ERD like structures:

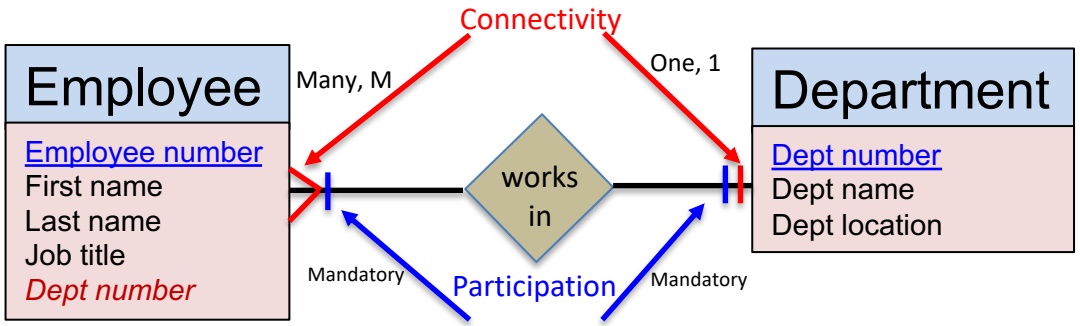


Summary of Keys

- **Super key**
  - Any key that uniquely identifies each row in an entity
- **Candidate key**
  - Minimal super key
- **Primary key**
  - Candidate key (chosen) to uniquely identify all other attributes in a given row
- **Secondary key**
  - Used only for data retrieval, cannot uniquely identity each row of the table
- **Composite key**
  - Key composed of more than one attribute
- **Key attribute**
  - Any attribute that is part of a key
- **Foreign key**
  - Values must match a primary key in a referenced (parent) table or be null

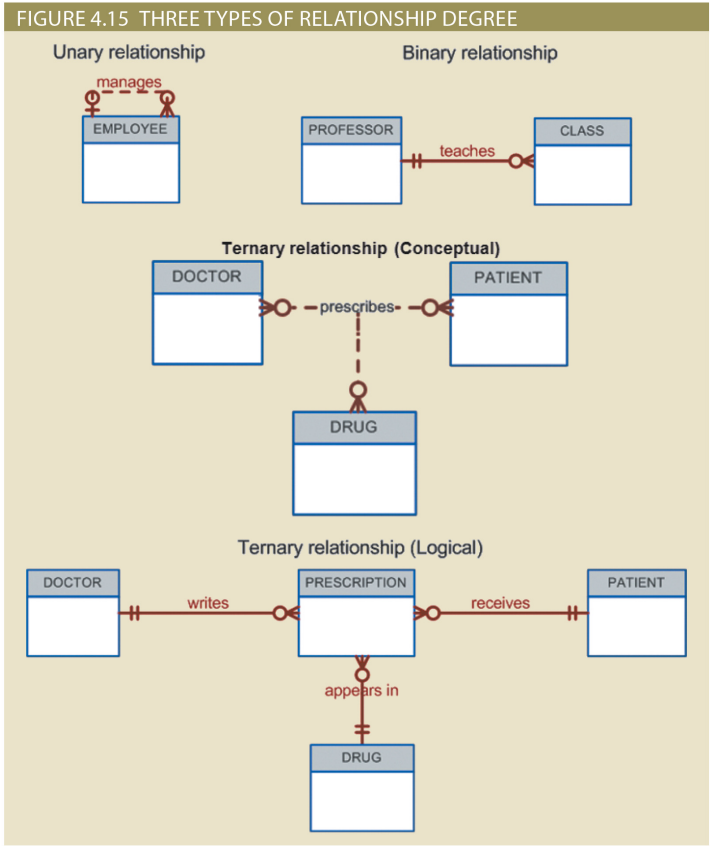
Relationship

- Connectivity specifies
  - The type of relationship between 2 entities: 1:1; 1:M; M:N
  - The maximum number of times an instance of an entity can be related to instances of the other entity in a relationship between 2 entities.
- Participation specifies the minimum number of times an instance of an entity can be related to instances of the other entity in a relationship between 2 entities: optional and mandatory.



Relationship Degree

- Indicates number of associated entities or participants
- **Unary relationship**
  - Association is maintained within a single entity
- **Binary relationship**
  - Two entities are associated
- **Ternary relationship**
  - Three entities are associated

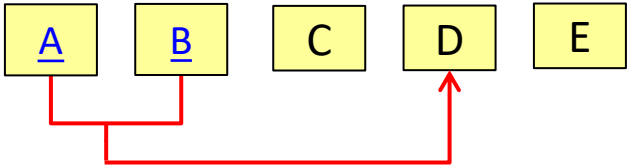


Dependency Summary

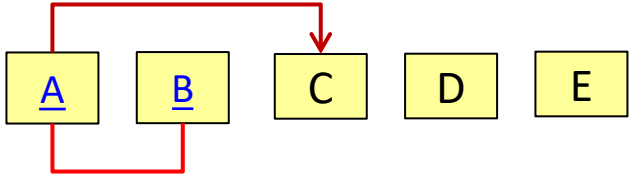
- **Functional**

- **Total**

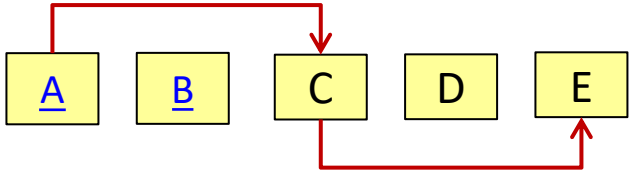
- **Full**



- **Partial**



- **Transitive**



Normalisation: From UNF to 3NF

Griffith UNIVERSITY

Conditions of 1NF:

- It is a valid relation (in particular, no null entries)
- A unique key (primary key) has been identified for each row
- All attributes are functionally dependent on all or part of the key

Conditions of 2NF:

- Relation is in 1NF and
- No partial dependency exists
  - So, if the primary key is a composite key, partial dependency may exist and we need to remove partial dependency to convert to 2NF
  - If the primary key is NOT a composite key, the table is already in 2NF!

Conditions of 3NF:

- Relation is in 2NF and
- No transitive dependency exists
  - So, if NO transitive dependency exists in a table, the table is already in 3NF!

Writing SQL Statement

Create a new database

- Syntax:

  CREATE DATABASE [IF NOT EXISTS] *db_name;*

- Example:

  CREATE DATABASE DB_Week7;

  or

  CREATE DATABASE **IF NOT EXISTS** DB_Week7;

- USE DB_Week7;

- SHOW TABLES;

- The IF NOT EXISTS option is useful if you are using a script to create a database. This determines if it already exists, if it does not the database is created.

- If you are doing this on the command line you should then use the USE command to indicate that you will be working with the new database.

- What does SHOW TABLES result?

Inserting Data into Tables

- The INSERT command is used to insert data into tables
- Can be:
  - One row at a time
  - Multiple rows
  - Using data from other tables

- **Insert – Syntax 1** (in all columns - no column listing required)

```
INSERT INTO tbl_name VALUES (val1,val2,...)
```

Staff table:

StaffID	StaffName	DateOfBirth	Salary
1	Buffy Summers	1987-09-15	27000.00

- Example:

```
INSERT INTO staff VALUES (NULL, 'Buffy Summers', '1987-09-15', 27000);
```

  - '**NULL**' here is a dummy place holder for the AUTONUMBER

UPDATE

Griffith UNIVERSITY

- **Example 1:** Update Staff Name details

```
UPDATE Staff
SET StaffName = 'Buffy Winters'
WHERE StaffId = 1
```

  - It updates the name for the staff with the id of 1!

Staff table:

StaffID	StaffName	DateOfBirth	Salary
1	Buffy Winters	1987-09-15	27000.00
2	Buffy Summers	1987-09-15	27000.00
3	Teddy Bear	1983-12-03	87125.02
4	John Smith	1972-09-20	25000.00
5	Jane Doe	1969-01-25	55000.00
6	Jacek Jones	1984-10-19	35000.00
7	Teddy Bear	1983-12-03	87125.02
8	Fred Smith	1956-06-30	25125.02

- **Example 2:** Update Staff Salaries up by 10%

```
UPDATE Staff
SET Salary = 1.1 * Salary;
```

  - It updates all staff salary by 10% increase!

Staff table:

StaffID	StaffName	DateOfBirth	Salary
1	Buffy Winters	1987-09-15	29700.00
2	Buffy Summers	1987-09-15	29700.00
3	Teddy Bear	1983-12-03	95837.52
4	John Smith	1972-09-20	27500.00
5	Jane Doe	1969-01-25	60500.00
6	Jacek Jones	1984-10-19	38500.00
7	Teddy Bear	1983-12-03	95837.52
8	Fred Smith	1956-06-30	27637.52

DELETE

- **Example 1:**

```
DELETE
FROM Staff
WHERE StaffID = 1;
```

  - It deletes all the records with the staff id of 1

Staff table:

StaffID	StaffName	DateOfBirth	Salary
2	Buffy Summers	1987-09-15	29700.00
3	Teddy Bear	1983-12-03	95837.52
4	John Smith	1972-09-20	27500.00
5	Jane Doe	1969-01-25	60500.00
6	Jacek Jones	1984-10-19	38500.00
7	Teddy Bear	1983-12-03	95837.52
8	Fred Smith	1956-06-30	27637.52

Staff with ID 1 no longer exists!

- **Example 2:**

```
DELETE
FROM Staff2
WHERE Salary > 80000;
```

  - It deletes all the records with the staff salary of more than 80000

Staff2 table:

StaffId	StaffName	DateOfBirth	Salary
1	Buffy Summers	1987-09-15	27000.00
2	Buffy Summers	1987-09-15	27000.00
4	John Smith	1972-09-20	25000.00
5	Jane Doe	1969-01-25	55000.00
6	Jacek Jones	1984-10-19	35000.00
8	Fred Smith	1956-06-30	25125.02

Staff with IDs 3 & 7 no longer exists!

ALTER: ADD | MODIFY | DROP column(s)

Griffith UNIVERSITY

- **ADD**

  ALTER TABLE Staff
  ADD Address VARCHAR(30);

  - It adds a new column to the table staff

- **MODIFY**

  ALTER TABLE Staff3
  MODIFY Salary INT(11);

  - It changes the data type of the column Phone

- **DROP**

  ALTER TABLE Staff
  DROP COLUMN Address;

  - It deletes the column ‘address’ from the table staff

Staff table:

StaffID	StaffName	DateOfBirth	Salary	Address
2	Buffy Summers	1987-09-15	29700.00	NULL
3	Teddy Bear	1983-12-03	95837.52	NULL
4	John Smith	1972-09-20	27500.00	NULL
5	Jane Doe	1969-01-25	60500.00	NULL
6	Jacek Jones	1984-10-19	38500.00	NULL
7	Teddy Bear	1983-12-03	95837.52	NULL
8	Fred Smith	1956-06-30	27637.52	NULL

+ Options

Staff3 table:

StaffID	StaffName	DateOfBirth	Salary
1	Buffy Summers	1987-09-15	27000
2	Buffy Summers	1987-09-15	27000
3	Teddy Bear	1983-12-03	87125
4	John Smith	1972-09-20	25000
5	Jane Doe	1969-01-25	55000
6	Jacek Jones	1984-10-19	35000
7	Teddy Bear	1983-12-03	87125
8	Fred Smith	1956-06-30	25125

Staff table:

StaffID	StaffName	DateOfBirth	Salary
2	Buffy Summers	1987-09-15	29700.00
3	Teddy Bear	1983-12-03	95837.52
4	John Smith	1972-09-20	27500.00
5	Jane Doe	1969-01-25	60500.00
6	Jacek Jones	1984-10-19	38500.00
7	Teddy Bear	1983-12-03	95837.52
8	Fred Smith	1956-06-30	27637.52

Address column does not exists anymore!

Update, delete data and drop table

- **Additional DML:**
  - UPDATE
  - DELETE

```
UPDATE Staff
SET Salary = 1.1 * Salary;
```

```
DELETE
FROM Staff2
WHERE Salary > 80000;
```

- **Additional DDL:**
  - DROP table
  - ALTER table
    - ADD column
    - MODIFY column
    - DROP column

```
DROP TABLE Staff2;
```

```
ALTER TABLE Staff
ADD  Address VARCHAR(30);
```

```
ALTER TABLE Staff3
MODIFY Salary INT(11);
```

```
ALTER TABLE Staff
DROP COLUMN Address;
```

- **Constraints**
  - Entity integrity
  - Referential integrity
  - NOT NULL
  - UNIQUE
  - DEFAULT
  - CHECK

```
CREATE TABLE IF NOT EXISTS DEPARTMENT(
        DepartmentID        INT PRIMARY KEY
AUTO_INCREMENT,
        DepartmentName      VARCHAR(30),
        Budget              DOUBLE,
        ManagerID           INT NOT NULL,
        FOREIGN KEY (ManagerID) REFERENCES
Staff(StaffID)
) ENGINE=InnoDB;
```

```
CREATE TABLE CUSTOMER (
CUS_CODE        NUMBER          PRIMARY KEY,
CUS_LNAME       VARCHAR(15)     NOT NULL,
CUS_FNAME       VARCHAR(15)     NOT NULL,
CUS_INITIAL     CHAR(1),
CUS_AREACODE    CHAR(3)         DEFAULT '615'    NOT NULL
                                CHECK(CUS_AREACODE IN
                                ('615','713','931')),
CUS_PHONE       CHAR(8)         NOT NULL,
CUS_BALANCE     NUMBER(9,2)     DEFAULT 0.00,
CONSTRAINT CUS_UI1 UNIQUE (CUS_LNAME, CUS_FNAME));
```

Select Statement

- **General Syntax**

```
SELECT [ALL | DISTINCT | DISTINCTROW ] select_expr [, select_expr ...]
FROM table_references
[WHERE where_condition]
[GROUP BY {col_name | expr | position} ]
[HAVING where_condition]
[ORDER BY {col_name | expr | position} [ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

CONCAT, CAST functions

- **CONCAT**
  - Merge columns in your table output: Show staff and their date of birth into one column.

```
SELECT CONCAT(StaffName, ' was born on ', DateOfBirth)
AS 'Staff date of birth'
  FROM Staff;
```

Staff date of birth
Buffy Winters was born on 1987-09-15
Teddy Bear was born on 1983-12-03
John Smith was born on 1972-09-20
Jane Doe was born on 1969-01-25
Jacek Jones was born on 1984-10-19
Mohammad Awrangjeb was born on 1977-11-21
Rupam Deb was born on 1980-10-21
Md Polash was born on 1981-11-25
Teddy Bear was born on 1983-12-03
Fred Smith was born on 1956-06-30

- **CAST**
  - Convert the data type: Change budget from DOUBLE to DECIMAL(9,2). Also, rename the budget column as ‘Annual Budget’ and show the output in descending order of the ‘Annual Budget’.

```
SELECT DepartmentName, CAST(Budget AS DECIMAL(9,2)) AS
'Annual Budget'
FROM DEPARTMENT
ORDER BY 'Annual Budget' DESC;
```

DepartmentName	Annual Budget
Sales	5005000.00
Marketing	509000.00
Finance	650000.00
Accounting	360000.00
Human Resource	550000.00

GROUP BY

- **How does it work?**
  - Make groups based on the column value: StaffID in this case
  - If the rows are not ordered, no problem: MySQL still group them
  - Apply, any conditions that come in **HAVING clause**!
  - Then, apply any aggregate function!

  - For example, for this SELECT statement:

```
SELECT StaffID, COUNT(DepartmentID) AS '# of department',
        SUM(PercentageTime) AS 'Total time fraction'
FROM workallocation
GROUP BY StaffID;
```

    - First, make groups based on staffID
    - No, HAVING clause, so nothing happen
    - Finally, apply COUNT() and SUM() functions!
    - So, the output looks like:

StaffID	# of department	Total time fraction
1	2	1
2	1	1
3	3	1
4	2	1
5	2	1
6	3	1
7	1	1
8	2	1
9	2	1
10	4	0.9

WorkAllocation table

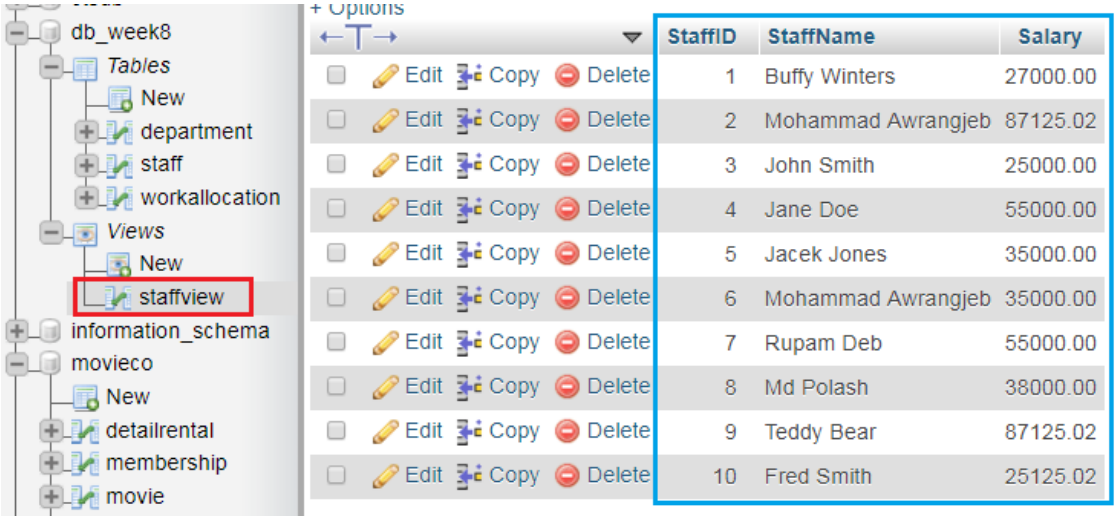
StaffID	DepartmentID	PercentageTime
1	2	0.7
1	5	0.3
2	1	1
3	2	0.3
3	3	0.2
3	4	0.5
4	4	0.3
4	5	0.7
5	3	0.7
5	4	0.3
6	3	0.4
6	4	0.3
6	5	0.3
7	5	1
8	2	0.4
8	3	0.6
9	4	0.5
9	5	0.5
10	1	0.4
10	3	0.2
10	4	0.2
10	5	0.1

View

Example:

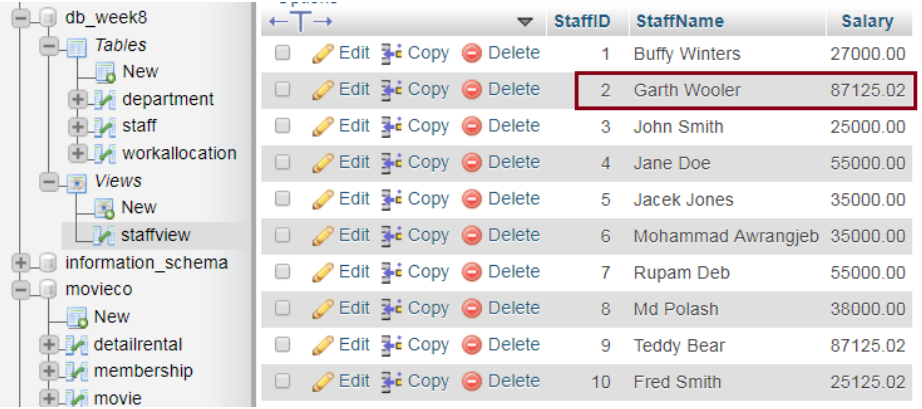
- For example, we create a view on staff table with staff name and salary

```
CREATE VIEW StaffView AS
SELECT StaffID, StaffName, Salary
FROM staff;
```



- Now change staff name with Staff

```
UPDATE staff
SET StaffName = 'Garth Wooler'
WHERE staffID = 2;
```



Subquery

- A subquery is a query that is embedded (or nested) inside another query
- Also known as a nested query or an inner query

- Syntax:

```
SELECT select_list
FROM   table
WHERE expr operator    (SELECT select_list
                       FROM   table);
```

Subquery with WHERE clause!

- The first query in the SQL statement is known as the outer query
- The query inside the SQL statement is known as the inner query
- The inner query is evaluated first and the output from this query is used as the input for the outer query
- The inner query is normally expressed inside parentheses

Final Exam Information

Final Exam Information

There are six questions in the final exam, question sets will be given to the students randomly:

- **Two** questions on designing ERD (one basic and one advance)
- **Two** questions on writing SQL statement (one basic and one advance)
- **One** question on Advance Topics
- **One** question on Normalization

You can find the Practice Final Exam on the course homepage

Thank you.