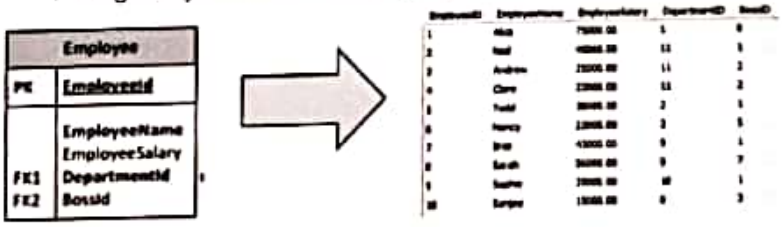
MOXUE EDUCATION INFO 90002 Database System & Information Modelling
Final Exam Revision

ER Diagram

- A database can be thought of as a representation of
  - a collection of entity sets, and
  - relationships between the entities
- An entity is an object or abstract concept or event which can be distinguished from other entities
  - Will have many instances in the database
  - Has several attributes
  - Is necessary for the system to work
  - example: product, order, sale, person, movie, tweet
- Entities have attributes that describe the entity and distinguish it from other entities in the same entity set
  - example attributes: EmployeeName, Address

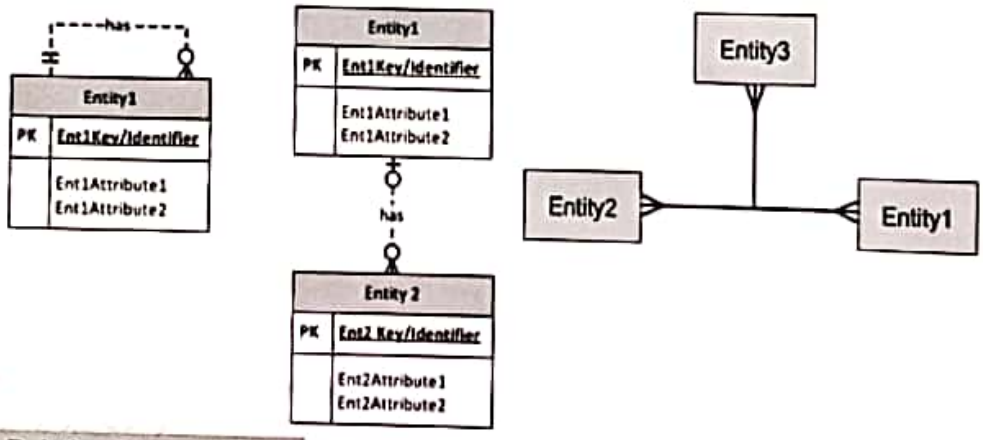
- Attributes: What things would we need to record about the entity
- Mapping ER diagram to database tables:

  - Entity set
    - Often corresponds to a table in the database
  - Entity instance
    - Often corresponds to a row in a table
  - Attribute
    - Often corresponds to a column in a table
  - Relationship set (link between entity sets)
    - Often corresponds to a Foreign Key in a table
  - Relationship instance (link between entity instances)
    - Foreign Key value = Primary Key value

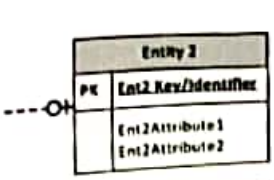


- **Rules:**
  - Underline = primary key
  - Dot line = foreign key
  - () = composite attributes
- **Keys:**
  - **Keys or Identifiers are used to identify individual entity instances**
    - **Primary Key**
      - (set of) columns, the values in which uniquely identify each instance
      - no column can be removed from the key without losing uniqueness
    - **Candidate Key**
      - the set of possible primary keys (choose one to be the PK)
    - **Surrogate Key**
      - system-assigned serial number (used if natural PK is unavailable or unsuitable)
    - **Composite Key**
      - a key which is made up of more than one attribute
        - **e.g. for the entity "airline flight" we might use the composite key**
          - **FlightNumber + FlightDate**
    - **Foreign Key**
      - the key used to link to a primary key in another table
        - **helps us to join tables in a Select statement**
  - **Primary Keys are**
    - **unique**
    - **never null**
    - **do not change their value**
- **Relationship degree:**

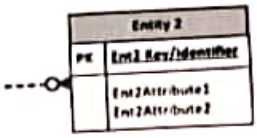
Unary (1) Binary (2) Ternary (3)



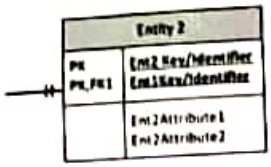
- **Relationship cardinality:**



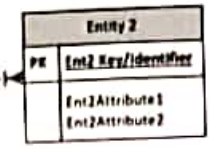
Optional One



Optional Many



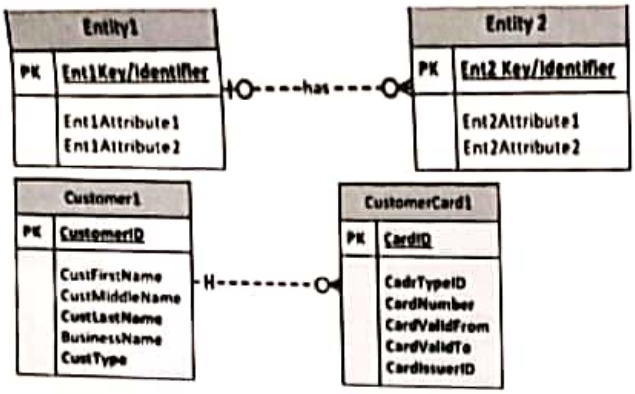
Mandatory One



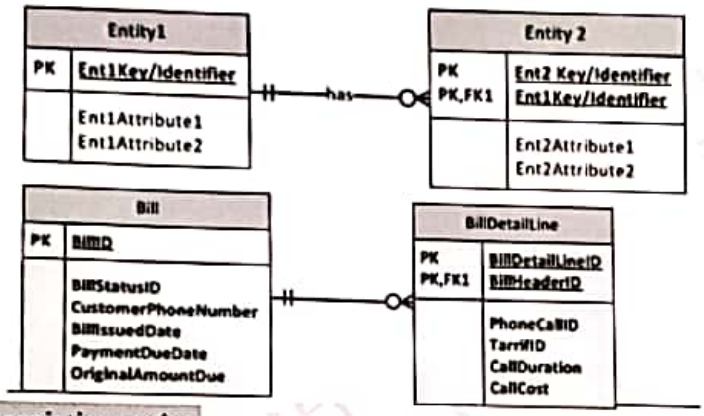
Mandatory Many

- 当关系为一(零)对多时，多的部分连接在表上，少的部分在相反方向。

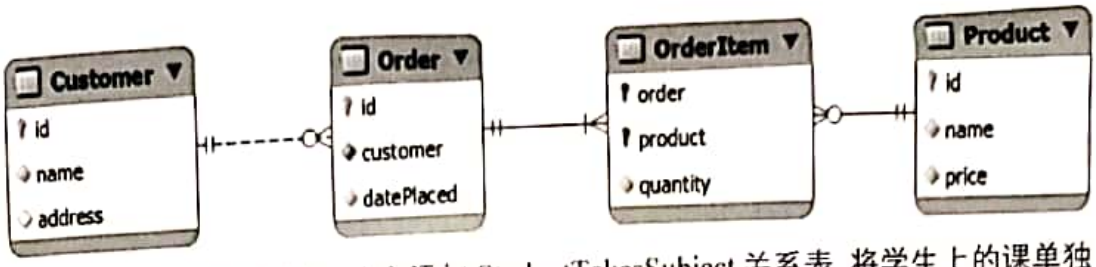
- **Binary relationships:**
  - One-to-Many
    - primary key on ONE side becomes foreign key on MANY side
  - Many-to-Many
    - create an Associative Entity (a new table) with a compound primary key consisting of 2 FKs that refer to the other 2 tables
    - you then have two One-to-Many joins
  - One-to-One
    - decide in which table to put the foreign key
    - foreign key on the optional side refers to primary key on the mandatory side
- **Unary relationships:**
  - "Unary" means there is only one table
    - the table is in a relationship with itself
    - (a row in the table can be linked to other rows in the same table)
  - Operate in the same way exactly as binary relationships
    - One-to-One
      - put a Foreign key in the entity
    - One-to-Many
      - put a Foreign key in the entity
    - Many-to-Many
      - create an extra table - Associative Entity
      - put two Foreign keys in the Associative Entity
        - the two FKs need different names
        - the FKs become the combined PK of the Associative Entity
- **Strong entity & Weak entity**
  - Strong entity: entity 2's PK is independent of the PKs of other entities
  - 用虚线连接关系



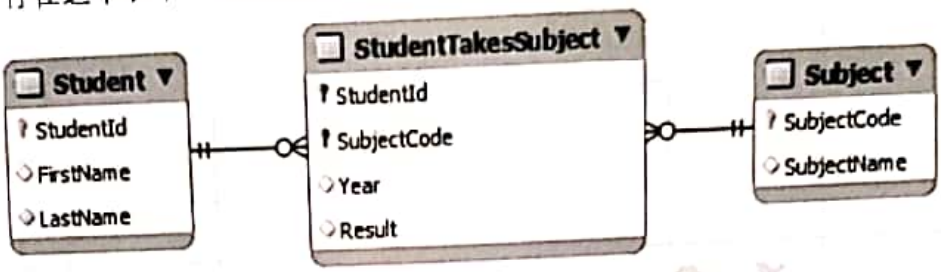
- Weak entity: entity 2's PK depends on (includes) the PK of entity 1
- 用实线连接关系



- Association entity:
  - When to create
    - when going from Conceptual to Logical phase of design
    - to implement a Many-to-Many relationship
    - to implement a Ternary relationship
  - The associative entity
    - has an independent meaning
    - has a unique identifier, usually a combination of FKs
    - may have attributes other than the FKs
    - may participate in other relationships
- 典型 ER 图例子
  - 关于订单系统: 注意添加 OrderItem 关系表, 将每个订单的物品及数量单独存在 OrderItem 中, 避免数据冗余



- 关于学生课程记录系统: 注意添加 StudentTakesSubject 关系表, 将学生上的课单独存在这个表中, 避免多对多的关系出现



SQL

- Consists of:
  - Data Definition Language (DDL)
    - to define and set up the database
    - CREATE, ALTER, DROP
      - also TRUNCATE, RENAME
  - Data Manipulation Language (DML)
    - to manipulate and read data in tables
    - SELECT, INSERT, DELETE, UPDATE
      - MySQL also provides others.... eg REPLACE
  - Data Control Language (DCL)
    - to control access to the database
      - GRANT, REVOKE
  - Other Commands
    - administer the database
    - transaction control
- **The SELECT statement:**

```
SELECT [ALL | DISTINCT] select_expr [, select_expr ...]
```
List the columns (and expressions) that are returned from the query

```
[FROM table_references
```
Indicate the table(s) or view(s) from where the data is obtained

```
[WHERE where_condition]
```
Indicate the conditions on whether a particular row will be in the result

```
[GROUP BY {col_name | expr } [ASC | DESC], ...]
```
Indicate categorisation of results

```
[HAVING where_condition]
```
Indicate the conditions under which a particular category (group) is included in the result

```
[ORDER BY {col_name | expr | position} [ASC | DESC], ...]
```
Sort the result based on the criteria

```
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```
Limit which rows are returned by their return order (ie 5 rows, 5 rows from row 2)

```
]
```

- Examples:
  - SELECT * FROM customer;
  - SELECT CustLastName, CustFirstName FROM customer;
  - SELECT CustLastName FROM customer where CustLastName = "Smith";
  - SELECT CustLastName FROM customer where CustLastName LIKE "Sm%";
  - SELECT CustLastName, CustTpye FROM customer ORDER BY CustLastName;
  - SELECT CustLastName, CustTpye FROM customer ORDER BY CustLastName DESC;
  - SELECT CustLastName, CustTpye FROM customer ORDER BY CustLastName LIMIT 5;
  - SELECT CustTpye, COUNT(CustomerID) FROM customer GROUP BY CustTpye;
  - SELECT CustTpye, COUNT(CustomerID) AS Count FROM customer GROUP BY CustTpye;
  - SELECT CustTpye, COUNT(CustomerID) AS Count FROM customer where CustLastName LIKE "Sm%" GROUP BY CustTpye;
  - SELECT CustTpye, COUNT(CustomerID) FROM customer where CustLastName LIKE "Sm%" GROUP BY CustTpye HAVING COUNT(CustomerID)=3;
- **Join:**

- Data about an entity is spread across 2 tables – so join them
- Inner/Equi join - Join rows where FK value = PK value

```
SELECT * FROM Customer INNER JOIN Account
    ON Customer.CustomerID = Account.CustomerID;
```

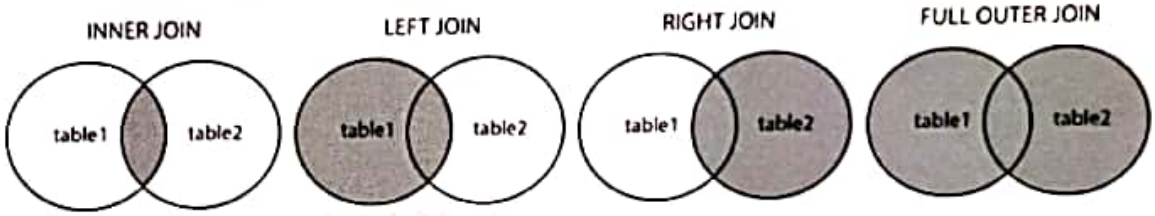
CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType	AccountID	AccountName	OutstandingBalance	CustomerID
1	Peter	NULL	Smith	NULL	Personal	1	Peter Smith	245.25	1
2	James	NULL	Jones	JJ Enterprises	Company	2	JJ Ent	552.39	2
2	James	NULL	Jones	JJ Enterprises	Company	3	JJ Ent Mgr	10.25	2

- Natural Join gives the same result as Inner Join
  - requires PK and FK columns to have the same name

```
SELECT * FROM Customer NATURAL JOIN Account;
```

CustomerID	CustFirstName	CustMiddleName	CustLastName	BusinessName	CustType	AccountID	AccountName	OutstandingBalance
1	Peter	NULL	Smith	NULL	Personal	1	Peter Smith	245.25
2	James	NULL	Jones	JJ Enterprises	Company	2	JJ Ent	552.39
2	James	NULL	Jones	JJ Enterprises	Company	3	JJ Ent Mgr	10.25

- **(INNER) JOIN:** Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN:** Return all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Return all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Return all records when there is a match in either left or right table



- Key words:
  - SELECT A FROM B: select 的基本语法, A 是要选择的列, B 是表
  - WHERE: 条件筛选, 比如可用>/</=来筛选数字范围
  - LIKE: 字符串模式匹配, %开头代表前面匹配任意字符, 后面一样; %结尾代表结尾匹配任意字符, 前面一样;%在中间代表字符串只要包含该字符即可
  - ORDER BY: 按某列排序, 默认升序
  - ASC/DESC: 升序或降序
  - GROUP BY: 按某一列分组, 一般和 COUNT()连用
  - HAVING: GROUP BY 后的条件筛选要用 HAVING
  - LIMIT: 取前几条数据
  - AS: 用于给列或表起别名。当多个表 JOIN 时为了方便写条件，或将输出的列改名方便阅读
  - AND/OR/NOT: 逻辑连接词

- A INNER JOIN B ON ...=...: 表 A 和表 B 内连接, 条件 PK=FK
- A LEFT JOIN B ON ...=...: 表 A 和表 B 左连接, 条件 PK=FK, 表 A 中没有对应的数据也会显示
- A RIGHT JOIN BON ...=...: 表 A 和表 B 右连接, 条件 PK=FK, 表 B 中没有对应的数据也会显示
- A NATURAL JOIN B: 不需要指定 PK 和 FK, 自行寻找列名相同的列进行对应。容易因重名出现错误, 不常用。
- IS NULL/IS NOT NULL: 某列为 null 或不为 null, 用来筛查空值
- UNION: 将两次返回结果并列起来, 需要结果的列相对应

- **Functions:**
  - COUNT(): 与 GROUP BY 一起用, 用于计数每组的数据条数。COUNT(*)代表计数所以数据行的总数包括空值, COUNT(某列)代表只计数这一列的数据行数
  - SUM(): 求某一行的总数
  - MAX(): 求某一行的最大值
  - MIN(): 求某一行的最小值
  - AVG(): 求某一行的平均值
  - FORMAT(N,D): N 是数值或某列数值, D 是要保留的几位小数, 返回结果是字符串, 不可以进行后续的排列
  - CAST(N AS DECIMAL(6,2)): 与 FORMAT()一样将数值保留为统一的几位小数, 但由于 CAST()可以将结果类型指定为小数, 因为可以进行后续排列
  - IFNULL(X,0/""): 把空值 null 转化为 0 或"", 便于计算和表达
  - LOWER()/UPPER(): 转换为全小写/全大写
  - LEFT()/RIGHT(): 取从左/右的几个字符
  - NOW(): 取当前系统时间
- Flow control:

```
SELECT employeeId, lastName, departmentId, salary,
CASE
    WHEN departmentId in
        (SELECT departmentId FROM Department WHERE name in ('clothes', 'books'))
    THEN salary * 0.2
    ELSE salary * 0.1
END as bonus
FROM employee
ORDER BY departmentid;
```

- **Subqueries:**