Strong Entity VS Weak Entity

Strong Entity
Has a key which may be defined without reference to other entities
某个 entity 里所有的 key 都来源于自己,不需要从别的 entity 中得到(conceptual 中的 key)

Weak Entity
Has a key which requires the existence of one or more other entities
-Key:数据库表中对储存数据对象予以唯一和完整标识某个 entity 的 key 是从其他 entity 中得到(conceptual 中的 key)

Entity (Strong Entity vs. Weak En4ty)
- Strong Entity (key 来源于自己)
- Weak Entity (key 从别的 Entity 拿来的)

Attribute
- Simple (gender, height…)
Cannot be subdivided 不能被进一步细分 e.g. age, gender...
- Composite(address, full name…)
Can be subdivided into additional attributes 可以进一步细分
- Single-Value (id, unit code…)
Can have only one value
Each product has one serial number 序列号
Single-valued NOT EQUAL TO simple attributes (But 可以并存)
e.g. serial number(composite & single-valued): CN-001-02-1234 (can be subdivide
- Multi-valued
Can have many values
e.g. degree (a person may have more than one degree)
- Derived (age, total price…)
Can be derived with algorithm
e.g. age can be calculated by dob

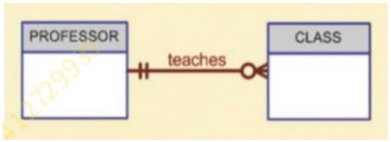
Relationship entity 和 entity 之间的关系
Identifying VS Non-Identifying
- Identifying
An Entity(A) supports another entity(B)
Use solid line (实线)
e.g. ENROLMENT 依赖于 STUDENT & UNIT 存在

- Non-Identifying
An Entity(A) does not supports another entity(B)
Use broken line (实线)
e.g. STUDENT , UNIT ,TUTOR 分别独立存在，不需要依赖对方

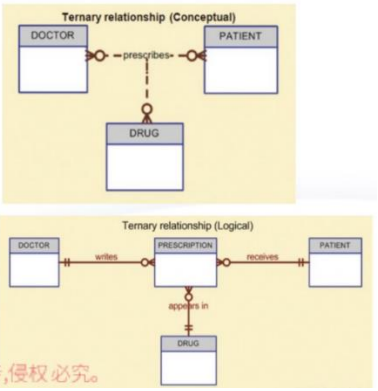
Relationship Connectivity
| : one
<: many
O: zero
| | : one and only one
|<: one to many(mandatory)强制
O|: zero or one(optional)可选择的
O<: zero to many(optional)可选择

Relationship Degree(多少个 entity 参与这段关系)
Unary(only one entity involved) 1

Binary(two entity involved) 2



Ternary(three entity involved)3



Relational Model
Property
- NO duplicate tuples 没有完全一样的两行数据
- Tuples have no order within a relation 表格中的行是无序的
- Attributes have no order 表格中的列是无序的
- Tuple values are atomic (multi-valued attributes are NOT allowed in the table)

Functional Dependency
通过 a 的值能得到一个且唯一一个 B 的值，例如 通过 student id 可以得到唯一一个 student name。
A->B: B is functionally depend on A, since the value of A determines a Single Value of B
A determines B, B depends on A(attribute 之间)

Key

Four types: Superkey, Candidate Key, Primary Key, Foreign Key

Super key: 能标识唯一 tuple 的属性的集合 有唯一性的 attribute 集合

{stud_id, stud_name} —> 唯一一行信息，所以{stud_id, stud_name} 是 superkey

{stud_id} —> 唯一一行信息，所以{stud_id} 是 superkey

{stud_name} —> 可能多于一行信息很多人都叫 jack，所以{stud_name} 不是 superkey

Candidate key：不含多余 attribute 的 superkey

- {stud_id, stud_name}是 superkey，但是输入 stud_name 自身实际上并不能保证只能得到唯一一行信息，所以 stud_name 属于 superkey 中多余的 attribute，stud_id 是 unique 的，可以保证只得到唯一一行信息，所以 stud_id 是 candidate key

- 同理，stud_oshc is candidate key (Every student has a unique stud_oshc)

Primary key: 从 candidate key 中选一个，每个表中有且仅有一个 pk (但有可能由多于一个的 attribute 组成 --> composite pk)

- better choose the one with numeric data type
- Natural pk(attribute 里有的) & Surrogate pk (logical model 自己写的)
- Natural: case 中本身带有的 - surrogate: 根据自自己己的需求添加的 (conceptual model 中不允许出现)

优先数字 primary key

Foreign key :

某个表的 fk 是另一表的 pk (两表有关联的前提下) 或为 NULL - 目的:描述两个表的关系

Data Integrity

Entity Integrity 实体完整性 primary key 唯一且不为空

- Primary key MUST NOT be NULL - Primary key MUST be UNIQUE

Referential Integrity 参照完整性

- The values of FK must either MATCH a value of full PK in the related relation or to be NULL
- FK 必须来自于某个有关联的表的 PK, 不然就为空

Column/Domain Integrity

- All values in a given column must come from the same domain (the same data type and range)

– 同一列的数据需要为同一数据类型以及范围

同一列的数据都是数字或者都是字母都是混合 范围都是在一个范围不能超出范围

Relational Algebra

Select σ选择运算 筛选条件

- Select rows (select one or more rows from [TABLE] where [CONDITION])

Project π 投影运算 筛选列 选择哪些列

- Select columns

Join 连接运算 当需要的数据不能从一个 table 得到

Theta-Join 选取属性间满足某条件的 tuples 进行连接

⋆Natural Join ⋈
选取带有相同意义属性间相同值的 tuples 进行连接(然后删除其中一个重复列) - 无法合并的行会被删除
Outer Join
Full Outer Join
根据选定的列进行合并，无论两个表的那一列是否存在相同的值
Left 保留左边选定列的所有数据，与右边进行连接
right 保留右边选定列的所有数据，与左边进行连接

SQL
SELECT
FROM
WHERE
GROUP by
HAVING
ORDER BY;

Query
Search condition
comparison
<, >, <=, >=, =, !=
Example: display employee_id which the employee salary is higher than 80,000

```
SELECT emp_id
FROM employee
WHERE emp_salary > 80000;
```

Range:
BETWEEN xxx AND xxx
Example: display employee id which the employee salary is between 10000 and 80000

```
SELECT emp_id
FROM employee
WHERE emp_salary BETWEEN 10000 AND 80000;
```

Search Condition (continue)
Set Membership: IN
To test whether the value of expression equals one of a set of values
city IN ('Melbourne','Sydney') 查找 city 这一列带有'Melbourne' 或 'Sydney'的行

Pattern Match: LIKE
To test whether a string matches a specified pattern
%: represents any sequence of zero or more characters _: represents any single character
example:
SELECT emp_id
FROM employees
WHERE emp_lname LIKE 'C% #找 Employee last name 以 C 开头的

other example:
%m: 以 m 结尾的
%abc%: 在任何位置带有 abc 的
_ _ C% : C 在第三位的

NULL:
IS NULL / IS NOT NULL: to test whether a column has a NULL value
Example:
SELECT *
FROM student
WHERE grade IS NULL;

Query(NVL, AS)
NVL(colname, value replace NULL) - in SELECT statement
NVL(enrol_mark, 0) 把 enrol_mark 为 NULL 的替换成 0 NVL(enrol_grade,'WH')

Rename: AS
SELECT stud_id, enrol_mark/10 AS new_mark FROM enrolment;
SELECT stud_id AS "student id" FROM ENROLMENT

Order by
Ascending(从小到大) & descending(从大到小)
不 写 desc 默认 asc

NULLS LAST & NULLS FIRST
NULLS LAST: 如果排序遇到 null 行，null 行放在最后
NULLS FIRST: 如果排序遇到 null 行，null 行放在最前

DISTINCT
如果出现重复的，只选择一行

SQL Query(case when then else end)

```
SELEC CASE
    WHEN  1  THEN  result 1
    WHEN  2  THEN  result 2
    ELSE  result 3
    END AS “”
```

Aggregate Function

```
Select sum(xxx)          SELECT avg()
From table               From table
                         Group by xxxx
```

Max() min() 最大最小值
Avg() 平均数
Count(*)输出行数

```
Select unit_id, count(*)     count(*)根据不同 unit_id 计算行数
From unit
Group by unit_id             group 分组
Having count(*)>2            筛选条件
```

Subquery
当需要的条件不在 select 里边，需要 subquery
Avg(mark)不在 u 子厚 select 的 output 里，但是需要他来对 date 进行比较
Nested：独立于外边的 query 只会执行一次
e.g

```
Select studid, unitcode, mark
From uni.enro
Where (unitcode,mark) IN (select unitcode, max(mark)
                          from uni.enro
                          group by unitcode)
Order by unitcode, studid;
```

Correlated：经过一次代码就执行一次，里边的 query 和外边的 query 有联系无法独立运行

```
Select studid, unitcode, mark
From uni.enro e1
Where mark = (select max(mark)
              From uni.enro e2
              Where e1.unitcode = e2.unitcode)
```

Inline: 通过计算得到，output 经常作为一个 table 跟在 from 后边

```
Select studid, e.unitcode, mark
From
```

(select unitcode, max(mark) as max_mark
From uni.enro
Group by unitcode) max_table
Join uni.enro e on e.unitcode = max_table.unitcode and
e.mark = max_tabel.max_mark

Query(View)
虚拟的 table 不能在 sql 里输出 用的时候有 不用的时候没有
Select unitcode,max(mark) as max_mark
From uni.enrolment
Group by unitcode
Select * from max_view
Order by unitcode

Query(relation set operator):

合并(join 左右合并两个表)
UnionAll：All rows selected by either query, including all duplicates **啥都不管直接合并**
Union：All rows selected by either query, removing duplicates (e.g,, DISTINCT on Union All)
去重合并
（上下合并 前提：attribute 必须一样）

交并集
Intersect： All distinct rows selected by both queries (找交集)
Minus：All distinct rows selected by the first query but not by the second (前表有后表没有的)
DB：Big Data
3V
Volume
Quality of data 数据量大

Velocity
Speed of processing data 处理速度要快

Variety
Variations in data 种类丰富
Structured, Unstructured, Semi-structured
Relational Database & Non-Relational Database

Relational Database：结构固定（table）
Relationship, Entity, Attribute 固定，管理方便 但是不灵活，存储数据类型局限
e.g Oracle SQL，MySQL

Non- Relationship：灵活（Document，graph，Column-base）

能存各种数据 data　能有各种各样的形式但是管理不规范，不固定，变化多需要适应和学习
e.g MongoDB(Documents), Cassandra(column-based), New4j (Graph)

Document
Each item is stored as a document (normallyBSON orJSON document, but could be XML)
每个项存储为一个文档(通常是 bson 或 json 文档，但也可以是 XML)
Note the variable structure and embedded documents
注意可变结构和嵌入文档

Column Family (also called Wide ColumnStore)
Key points to a set of multiple column values containing related data arranged by column family 包含按列族排列的相关数据的一组多列值的关键点

Graph - based on a graph structure
Unlike the previous three which are aggregation oriented, the graph model views data at a highly non aggregated level Based on graph theory
与前三种面向聚合的图模型不同，图模型在高度非聚合的级别上查看数据
Navigate via relationships (edges) between nodes
Examples
Neo4j　HyperGraphDB

Document
Key : value
Composite: {key: value, key: value}
Mult-valued:1. key: [value1, value2, value3]
2. key: [{1},{2},{3}]

Document json

String vs number vs boolean

mongoDB CRUD(CREATE, RETRIEVE, UPDATE, DELETE)
Create:
InsertOne: 只 insert 一条记录
insertMany: insert 多条记录（要用 array）
Create collection by inserting documents
db.collection.insertOne(…JSON…);

db.collection.insertMany - insert an array of JSON documents.
insertMany ([JSON1, JSON2, ...]);

RETRIEVE:
Find()
Documents retrieved by find method on collection
db.famous.find({}); or db.famous.find({}).pretty()
find all
$and: db.famous.find({$and:[{cond1},{cond2}..})
$or: db.famous.find({$or:[{cond1},{cond2}..})
.count() db.famous.find().count()
.sum() db.famous.find().sum()
.pretty() db.famous.find().pretty()
db.famous.find ([predicate));. find according to predicate, count() to count instances

UPDATE
updateOne: 只 update 一条记录
updateMany: update 多条记录
db.famous.updateOne ({condition},{$set:output});

If update array: “enrolment.$.unit”:”FIT1008”符合条件的情况下只更新第一个 element 的值
$push:更新 array 信息{$push: {“skills”:{“code”:”python”,”level”:4}}}
$set:替换数据 $push：插入一条新的

DELETE
deleteOne: 只 delete 一条记录
deleteMany: delete 多条记录
db.famous.deleteOne ({condition});

Transaction Management

Arithmetic Functions
Abs(n)计算绝对值 |a|
The column's absolute value
select abs(sallower - salupper) from salgrade;

ceil(n)大于或等于 number 的最接近的整数 2.1 = 3 3.1 = 4
Nearest whole integer greater than or equal to number
select ceil(10.6) from dual;

floor(n)等于或小于 n 的最大整数 2.9 = 2 3.1 = 3
Largest integer equal to or less than n
select floor(10.6) from dual;

mod(m,n) m/n 的余数 7/5 = 1…2 return 2

Remainder of m divided by n. If n=0, then m is returned
select mod(7,5) from dual;

power(m,n) m^n m 的 n 次方
Number m raised to the power of n
select power(3,2) from dual;

round(n,m) 四舍五入到小数点 m 位
Results rounded to m places to the right of decimal point
select round(15.193,1) from dual;

sign(n) n=0=0 n>0 = 1 n<0 = -1
If n=0, returns 0; if n>0, returns 1; if n<0, returns -1
select sign(12 - 45) from dual;

sqrt(n) n 的二次方
Square root of n
select sqrt(120) from dual;
select round(sqrt(120),2) from dual;

trunc(n,m) 截取到几位小数 不算四舍五入
Truncates n to m decimal points, if m is omitted then n is truncated to 0 places
select trunc(15.79,1) from dual;
select trunc(15.79) from dual;

Text Functions
initcap(char) 每个字符串的第一个字符改为大写
Changes the first character of each character string to uppercase
select initcap('mr teplow') from dual;

lower(char), upper(char) 字符改为大写
Makes the entire string lowercase/uppercase
select lower(ename) from employee;

replace(char, str1, str2) 每次出现 str1 都会被替换为 str2
Character string with every occurrence of str1 being replaced with str2
select replace('jack and jue','j','bl') from dual;

substr(char,m,n) 从 m 开始截取 n 个字符
Picks off part of the character string char starting in position m for n characters
select substr('ABCDEF',2,1) from dual;

length(char) 字符长度
Length of char

```
select length('Anderson') from dual;

str1 || str2 1+2
Concatenates two character fields together
select deptname || ', ' || deptlocation as "Department Name and
Location"
from department;

lpad(char,n,char2)/rpad(char,n,char2) 从左或从右用 char2 将 char 填充到 n 个字符
Pads char left/right to size n using char2
select lpad('Page 1', 15, '*') as "Lpad example"
from dual;
select rpad('Page 1', 15, '*') as "Rpad example"
from dual;

ltrim(char[, k]), rtrim(char[, k])
remove characters from the left/right of char, until the first character not in k - if k is
not specified blanks are trimmed
select ltrim('Intro to SQL', 'InorSt ') from dual;

trim(char) 从 char 中删除开头和结尾的空格(空格)
remove leading and trailing blanks (spaces) from char
select trim(' Intro to SQL ') from dual;
```