**Ex-4 ERD Solutions**

Note: Table names in blue are added from converting a many to many relationship. Blue attributes in those tables are foreign keys.

Attributes in blue are foreign keys added from converting 1-n relationships. Attributes in red are attributes added from converting relationship attributes of 1-n relationships. Red is also used to indicate changes in some problems.

**Problem 1: Unique Wholesale**



Here are the tables:

Order: (Order#, OrderDate, Vendorid) (VendorId is a foreign key.)

Product: (Stock#, Price, Desc)

Vendor: (VendorId, Vname, address, phone)

Payment: (PaymentNo, Pdate, Amount, Order#)

OrderDetails: (Order#, Stock#) (Order# and Stock# are both foreign keys)

Note: VName here is company name.

**Problem 2:ValleyClub Dinner Events**



Member: (memid, FName, Lname, Address, Phone)

DinnerEvent: (EventId, Edate, location, time, Dessertid)

Entrée: (EntréeId, EntréeName, Desc)

Dessert: (DessertId, Dessertname, Desc)

EntréeChoices: (EvenetId, EntréeID)

MemberAttendsDinner: (memid, eventId)

**Note:** This is assuming database needs to store the description of each dessert and of each entrée. This does not store which entrée each member chose for each dinner event.

**Problem 3: InDisguise Oil Change**



Vehicle: (plate#, make, model, year, custid)

Customer: (custid, custname, address, phone)

Transaction: (tranid, trandate, $Total, gradename, plate#, price/qt, amount)

Oil: (gradename, price/QT)

Part: (partid, partname, partcost, laborcost)

PartsReplaced: (tranid, partNum, qty, UnitPartCost, UnitLaborCost)

**Problem 4**: Prepare an E-R diagram for a real estate firm that lists properties for sale.

First, we’ll leave out “reporting to manager” requirement since it requires a unary relationship, which we didn’t talk about.



Employee: (employeeid, ename, firstDate, birthdate, office#, ReportsToEmpId)

Property: (Pid, address, ListingOffice#)

Owner: (oid, address, phone, name)

Office: (office#, address, phone, ManagingEmployeeId)

PropertyOwnerShipInfo: (PId, Oid, %ownership)

Note the problem did not ask us to record which employee has the listing of a property, only the office that has the listing.

The above solution does not document the requirement that database needs to store each employee’s manager (supervisor). Documenting this requirement requires a unary relationship: a relationship between an entity and itself. See Solution below indicated in red.



Employee: (employeeid, ename, firstDate, birthdate, office#, superviserId) superviseId is a fk.

Property: (Pid, address, ListingOffice#)

Owner: (oid, address, phone, name)

Office: (office#, address, phone, ManagingEmployeeId)

PropertyOwnerShipInfo: (PId, Oid, %ownership)

**Problem 5:**Design a database for *the sales function* of a bookstore so that they can use it for sales.



Customer: (cid, cname, address, phone)

Order; (Order#, Ordate, , #copies, cid, ISBN)

Transaction: (tranid, trandate, cid)

Book: (ISBN, title, price, cover, #pages, qtyStock, qtyOrder, catname)

Author: (lastname, firstname)

Category: (name, location)

TransactionDetails: (tranid, ISBN, qty)

BookAuthorInfo: (ISBN, FirstName, Lastname)

Note: if you did not specify #copies as an attribute of Order entity, that is fine. Then the final table Order should not contain #copies either.

**Problem 6** Design an ERD that captures information needed for the Hillside county Basketball Conference (HCBC) (an amateur basketball association).



**Note:** If the database designer assigns a database-specific ID to each player that is unique across the entire league, then PlayerID (or PlayerNum) is the primary key for the player entity. In that case, the database needs to also store the Jersey number of each player. This maybe the preferred solution because it is easy to modify the ERD to accommodate history—a player may move to another team and get a different Jersey number but that player’s PlayerId would not change.

The solution presented here uses a player’s jersey number to identify each player. Since two players on different teams can have the same jersey number, jersey number does not uniquely identify each player. Jersey number + teamName uniquely identifies each player. So teamName needs to be added as part of the primary key. Since the primary key of another entity is needed to serve as the primary, Player entity is called a **Weak** Entity.

The disadvantage of using a weak entity for Player entity is that if later on the users say that they want to record history of each player, it is not a good idea to do a history entity for the primary key attributes. You would then revert to the first option discussed above and add a history entity to capture player movements between teams.

**Problem 7** Library



Student: (SSN, LN, FN, address, phone

Book: (Barcode, ISBN, Title, CoverType, BookType)

BookType: (Tname, LengofLoan

Transaction: (TranId, TranDate/Time, SSN)

TransactionDetails: (TranId, Barcode, DueDate, ReturnDate)

**Problem 8** Fundraising



Organization: (OName, phone, Headquarter, address, CharityType)

Donor: (DonorId, LName, FName)

Type: (TypeId, Desc)

Event: (EventId, Date, Time, Location, CharityType, OrgEmpId)

EventParticipants: (OName, EventId)

Donation: (EventId, DonorId, Amount)

**Problem 9**: Design a database for the sales function of a online order clothing company.



Tables:

ShippingCharge: (ChargeCode, fromValue, toValue, charge)

Order: (Order#, OrdDate, card#, CardType, expirationDate, OrderTotal, ChargeCode, cid, shippingCharge)

Customer: (cid, cname, address, phone)

Clothes: (CatNum, StyleNo, Size, Color, Price, Desc, UnitsAvailable)

OrderDetails: (Order#, CatNum, quantity)

Note: attributes such as CustomerName and Address are called aggregate or composite attributes. These are attributes that eventually need to be broken down further. For example, CustomerName can be broken down into CLName, CFName and address will be broken down into different components of an address (number, street, etc.).

**Problem 10: Dental office visits**.



Patient: (pid, pname, address, phone, dentistid, insuranceCoName)

Dentist: (did, dname)

Visit: (visitid, vdate, patientid, dentistid)

Insurnaceco: (ComName, address, phone)

Service: (Same, price)

VisitDetails: (visitId, servicename, price)

InsuranceCoverage: (insuranceName, ServiceName, Limit)

**Problem 10: Dental office visits**, including all history entities. (History Entity concept is not Required for this class.)

 Patient: (pid, pname, address, phone, dentistid, ~~insuranceCoName~~) FK: dentistid

(InsuranceCoName is deleted from here and replaced with CareHistory table)

CareHistory: (HID, st, end, DentistId, PatientId) FKs: DentistId, PatientId

Dentist: (did, dname)

Visit: (visitid, vdate, patientid, dentistid) FKs: patientId, dentistid

Insurnaceco: (Iname, address, phone)

Service: (Sname~~, price~~)

(Price is deleted from here and replaced with PriceHistory Table)

PriceHistory: (HID, st, End, Price, SName) FK: SName

VisitDetails: (visitId, servicename, price) FKs: VisitId, ServiceName

~~InsuranceCoverage: (insuranceName, ServiceName, Limit)~~

This table is deleted and replaced by the coverageHistory table

CoverageHistory: (HID, St, End, Limit, InsuranceName, ServiceName)

FKs: InsuranceName, ServiceName

**Problem 11: Shopping Mall Income Management**



Retailer: (RName, HQPhone, HQAddress)

Mall: (MallName, Address, Phone)

Shop: (ShopId, flatcharge, %ofRevenue, RetailerName, MallName)

MonthlySalesRevenue: (ShopId, MonthYr, SalesRev, MonthlyRent)

Note: this solution does not keep historical store rental data. If the shop is rented to a different retailer at a later time, only the current retailer of the shop is recorded in shop table and current rent structure (flatcharge and %ofRev).

The following is an alternative solution, converting this ERD to tables would give you the same tables. Revenue here is really a history entity. It stores not only revenue of the current month, but also of prior months.



Retailer: (RName, HQPhone, HQAddress)

Mall: (MallName, Address, Phone)

Shop: (ShopId, flatcharge, %ofRevenue, RetailerName, MallName)

Revenue: (ID, ShopId, MonthYr, SalesRev, MonthlyRent)

In Revenue table, ID can be deleted and shopid+monthYr used as the PK instead.

**Problem 12: Pet Store**



**Problem 13: Barbara’s Booking Reading Competition**



**Student: SID, LName, FName**

**Competition: compId, StartDate, EndDate, WinnerID(Sid), NumBooks**

**Book: ISBN**, Title

**Theme: Theme**

**ThemeBook: Theme, ISBN**

**CompetitionTheme: compId, Theme,**

**Competition Books: compId, ISBN**

**Problem 14**:



Customer: (Cust#, CName, Street, City, State, Zip)

WorkOrder: (OrderId, Cust#, CreateDate, RequiredDate, CompletionDate, Street, City, State, City)

Task: (Task#, TName, Hrly-Rate)

WorkOrderTask: (OrderId, Task#, StartDate, FinishDate, EstHrs, ActualHrs)

**Problem 15**



There are two relationships between ServiceLevel and Channel: one is “contains” and the other is “Requires minimally.” The former is m-n and the latter is 1-n. Either of them is fine. But do not use both.

Here, I chose to represent “requires minimally.” This is because we only need to store the lowest level that a channel is available. A higher level includes all channels offered at lower levels.

**Problem 16**



**Problem 17: Revising ValleyClub Dinner Event**



**Problem 18: Revising InDisguise Oil Change problem.**

