CSE 204 - INTRO TO DATABASE SYSTEMS PHYSICAL DATABASE DESIGN

Joseph LEDET
Department of Computer Engineering
Akdeniz University
josephledet@akdeniz.edu.tr

OUTLINE

- Purpose of physical database design.
- How to map the logical database design to a physical database design.
- How to design base relations for target DBMS.
- How to design general constraints for target DBMS.
- How to select appropriate file organizations based on analysis of transactions.
- When to use secondary indexes to improve performance.
- How to estimate the size of the database.
- How to design user views.
- How to design security mechanisms to satisfy user requirements.

LOGICAL V. PHYSICAL DATABASE DESIGN

- Sources of information for physical design process includes logical data model and documentation that describes model.
- Logical database design is concerned with the what, physical database design is concerned with the how.

PHYSICAL DATABASE DESIGN

- Process of producing a description of the implementation of the database on secondary storage.
- It describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.

OVERVIEW OF PHYSICAL DATABASE DESIGN METHODOLOGY

- Step 3 Translate logical data model for target DBMS
  - Step 3.1 Design base relations
  - Step 3.2 Design representation of derived data
  - Step 3.3 Design general constraints
- Step 4 Design file organizations and indexes
  - Step 4.1 Analyze transactions
  - Step 4.2 Choose file organizations
  - Step 4.3 Choose indexes
  - Step 4.4 Estimate disk space requirements

OVERVIEW OF PHYSICAL DATABASE DESIGN METHODOLOGY

- Step 5 Design user views
- Step 6 Design security mechanisms
- Step 7 Consider the introduction of controlled redundancy
- Step 8 Monitor and tune operational system

STEP 3 TRANSLATE LOGICAL DATA MODEL FOR TARGET DBMS

To produce a relational database schema from the logical data model that can be implemented in the target DBMS.

- Need to know functionality of target DBMS such as how to create base relations and whether the system supports the definition of:
  - PKs, FKs, and AKs;
  - required data – i.e. whether system supports NOT NULL;
  - domains;
  - relational integrity constraints;
  - general constraints.

STEP 3.1 DESIGN BASE RELATIONS

To decide how to represent base relations identified in logical model in target DBMS.

- For each relation, need to define:
  - the name of the relation;
  - a list of simple attributes in brackets;
  - the PK and, where appropriate, AKs and FKs.
  - referential integrity constraints for any FKs identified.

STEP 3.1 DESIGN BASE RELATIONS

- From data dictionary, we have for each attribute:
  - its domain, consisting of a data type, length, and any constraints on the domain;
  - an optional default value for the attribute;
  - whether it can hold nulls;
  - whether it is derived, and if so, how it should be computed.

DBDL FOR THE PROPERTYFORRENT RELATION

```
    Domain PropertyNumber:          variable length character string, length 5
    Domain Street:                  variable length character string, length 25
    Domain City:                    variable length character string, length 15
    Domain Postcode:                variable length character string, length 8
    Domain PropertyType:            single character, must be one of 'B', 'C', 'D', 'E', 'F', 'H', 'M', 'S'
    Domain PropertyRooms:           integer, in the range 1–15
    Domain PropertyRent:            monetary value, in the range 0.00–9999.99
    Domain OwnerNumber:             variable length character string, length 5
    Domain StaffNumber:             variable length character string, length 5
    Domain BranchNumber:            fixed length character string, length 4
PropertyForRent(
    propertyNo      PropertyNumber      NOT NULL,
    street          Street              NOT NULL,
    city            City                NOT NULL,
    postcode        Postcode,
    type            PropertyType        NOT NULL DEFAULT 'F',
    rooms           PropertyRooms       NOT NULL DEFAULT 4,
    rent            PropertyRent        NOT NULL DEFAULT 600,
    ownerNo         OwnerNumber         NOT NULL,
    staffNo         StaffNumber,
    branchNo        BranchNumber        NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (ownerNo) REFERENCES PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
                                    ON UPDATE CASCADE ON DELETE NO ACTION,
    FOREIGN KEY (branchNo) REFERENCES Branch(branchNo)
                                    ON UPDATE CASCADE ON DELETE NO ACTION);
```

STEP 3.2 DESIGN REPRESENTATION OF DERIVED DATA

To decide how to represent any derived data present in logical data model in target DBMS.

- Examine logical data model and data dictionary, and produce list of all derived attributes.
- Derived attribute can be stored in database or calculated every time it is needed.
- Option selected is based on:
  - additional cost to store the derived data and keep it consistent with operational data from which it is derived;
  - cost to calculate it each time it is required.
- Less expensive option is chosen subject to performance constraints.

PROPERTYFORRENT RELATION AND STAFF RELATION WITH DERIVED ATTRIBUTE NOOFPROPERTIES

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Staff

staffNo	fName	lName	branchNo	noOfProperties
SL21	John	White	B005	0
SG37	Ann	Beech	B003	2
SG14	David	Ford	B003	1
SA9	Mary	Howe	B007	1
SG5	Susan	Brand	B003	0
SL41	Julie	Lee	B005	1

STEP 3.3 DESIGN GENERAL CONSTRAINTS

To design the general constraints for target DBMS.

- Some DBMS provide more facilities than others for defining enterprise constraints. Example:

```
CONSTRAINT StaffNotHandlingTooMuch
CHECK (NOT EXISTS (SELECT staffNo
FROM PropertyForRent
GROUP BY staffNo
HAVING COUNT(*) > 100))
```

STEP 4 DESIGN FILE ORGANIZATIONS AND INDEXES

To determine optimal file organizations to store the base relations and the indexes that are required to achieve acceptable performance; that is, the way in which relations and tuples will be held on secondary storage.

- Must understand the typical workload that database must support.

STEP 4.1 ANALYZE TRANSACTIONS

To understand the functionality of the transactions that will run on the database and to analyze the important transactions.

- Attempt to identify performance criteria, such as:
  - transactions that run frequently and will have a significant impact on performance;
  - transactions that are critical to the business;
  - times during the day/week when there will be a high demand made on the database (called the peak load).
- Use this information to identify the parts of the database that may cause performance problems.
- Also need to know high-level functionality of the transactions, such as:
  - attributes that are updated;
  - search criteria used in a query.

STEP 4.1 ANALYZE TRANSACTIONS

- Often not possible to analyze all transactions, so investigate most 'important' ones.
- To help identify these can use:
  - transaction/relation cross-reference matrix, showing relations that each transaction accesses, and/or
  - transaction usage map, indicating which relations are potentially heavily used.
- To focus on areas that may be problematic:
  1. Map all transaction paths to relations.
  2. Determine which relations are most frequently accessed by transactions.
  3. Analyze the data usage of selected transactions that involve these relations.

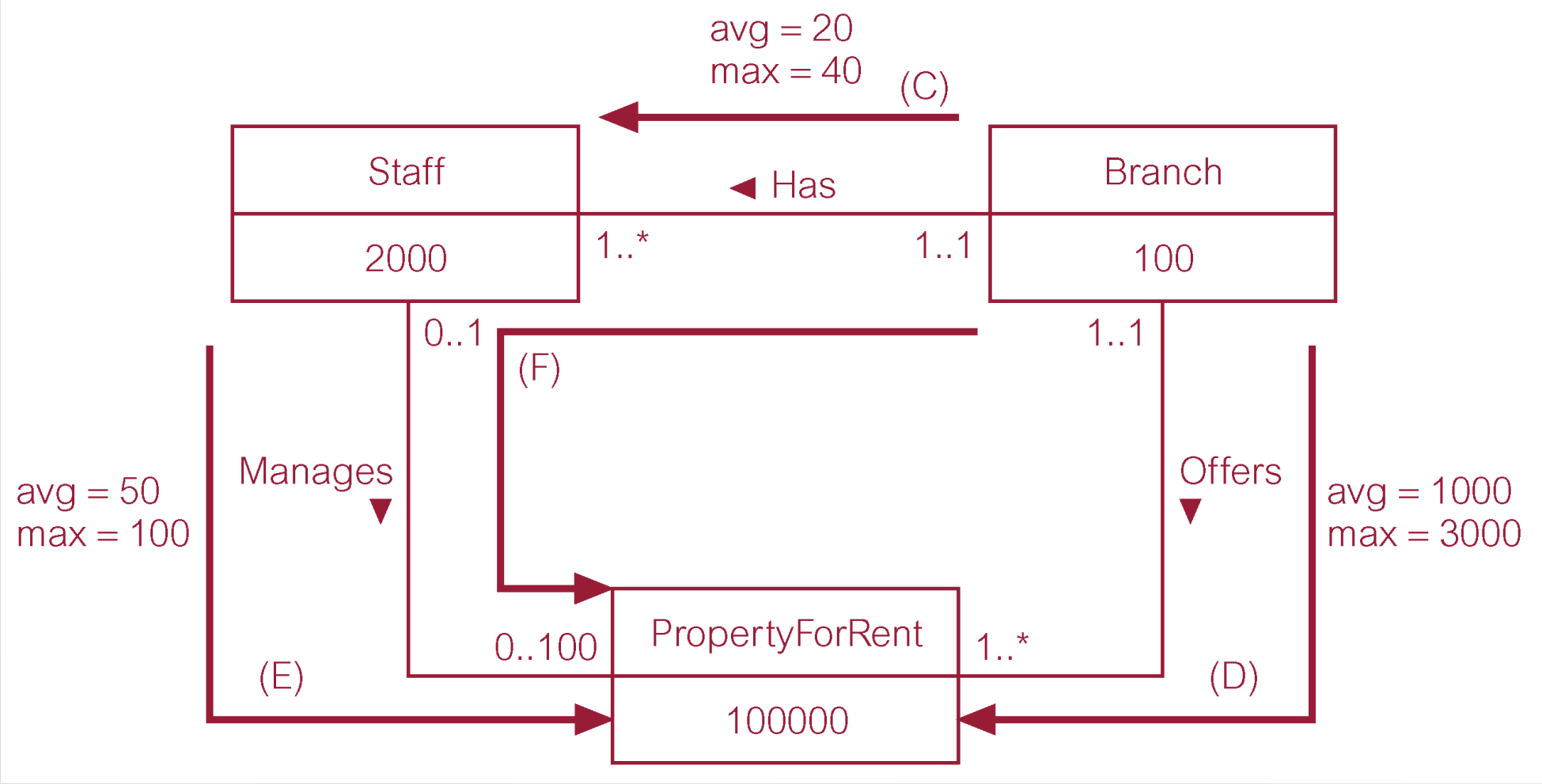
CROSS-REFERENCING TRANSACTIONS AND RELATIONS

Table 17.1 Cross-referencing transactions and relations.

Transaction/ Relation	(A)				(B)				(C)				(D)				(E)				(F)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch										X				X								X		
Telephone																								
Staff		X				X				X								X				X		
Manager																								
PrivateOwner	X																							
BusinessOwner	X																							
PropertyForRent	X					X	X	X						X				X				X		
Viewing																								
Client																								
Registration																								
Lease																								
Newspaper																								
Advert																								

I = Insert; R = Read; U = Update; D = Delete

EXAMPLE TRANSACTION USAGE MAP



EXAMPLE TRANSACTION ANALYSIS FORM

Transaction Analysis Form **1-Sept-2004**

Transaction (D) List the property number, address, type, and rent of all properties in Glasgow, ordered by rent

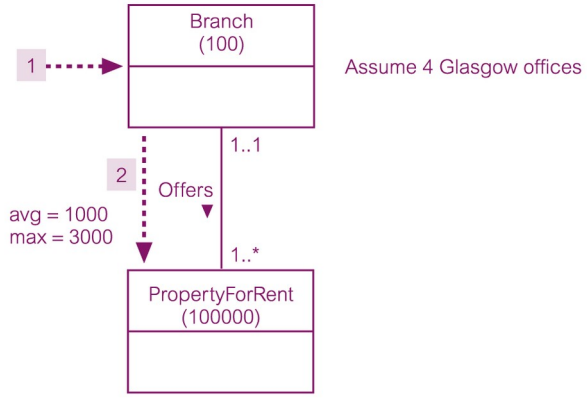
Transaction volume

Average: 50 per hour
Peak: 100 per hour (between 17.00 and 19.00 Monday–Saturday)

```
SELECT propertyNo, p.street, p.postcode, type, rent
FROM Branch b INNER JOIN PropertyForRent p ON
     b.branchNo = p.branchNo
WHERE p.city = 'Glasgow'
ORDER BY rent;
```

Predicate: p.city = 'Glasgow'
Join attributes: b.branchNo = p.branchNo
Ordering attribute: rent
Grouping attribute: none
Built-in functions: none
Attributes updated: none

Transaction usage map



Access	Entity	Type of Access	No. of References		
			Per Transaction	Avg Per Hour	Peak Per Hour
1	**Branch** (entry)	R	100	5000	10000
2	**PropertyForRent**	R	4000–12000	200000–600000	400000–1200000
Total References			**4100–12100**	**205000–605000**	**410000–1210000**

Figure 17.4 Example transaction analysis form.

STEP 4.2 CHOOSE FILE ORGANIZATIONS

To determine an efficient file organization for each base relation.

- File organizations include Heap, Hash, Indexed Sequential Access Method (ISAM), B+-Tree, and Clusters.
- Some DBMSs may not allow selection of file organizations.

STEP 4.3 CHOOSE INDEXES

To determine whether adding indexes will improve the performance of the system.

- One approach is to keep tuples unordered and create as many **secondary indexes** as necessary.
- Another approach is to order tuples in the relation by specifying a primary or clustering index.
- In this case, choose the attribute for ordering or clustering the tuples as:
  - attribute that is used most often for join operations - this makes join operation more efficient, or
  - attribute that is used most often to access the tuples in a relation in order of that attribute.

STEP 4.3 CHOOSE INDEXES

- If ordering attribute chosen is key of relation, index will be a primary index; otherwise, index will be a clustering index.
- Each relation can only have either a primary index or a clustering index.
- Secondary indexes provide a mechanism for specifying an additional key for a base relation that can be used to retrieve data more efficiently.

STEP 4.3 CHOOSE INDEXES

- Have to balance overhead involved in maintenance and use of secondary indexes against performance improvement gained when retrieving data.
- This includes:
  - adding an index record to every secondary index whenever tuple is inserted;
  - updating secondary index when corresponding tuple updated;
  - increase in disk space needed to store secondary index;
  - possible performance degradation during query optimization to consider all secondary indexes.

STEP 4.3 CHOOSE INDEXES – GUIDELINES FOR CHOOSING 'WISH-LIST'

1. Do not index small relations.
2. Index PK of a relation if it is not a key of the file organization.
3. Add secondary index to a FK if it is frequently accessed.
4. Add secondary index to any attribute heavily used as a secondary key.
5. Add secondary index on attributes involved in: selection or join criteria; ORDER BY; GROUP BY; and other operations involving sorting (such as UNION or DISTINCT).

STEP 4.3 CHOOSE INDEXES – GUIDELINES FOR CHOOSING 'WISH-LIST'

6. Add secondary index on attributes involved in built-in functions.
7. Add secondary index on attributes that could result in an index-only plan.
8. Avoid indexing an attribute or relation that is frequently updated.
9. Avoid indexing an attribute if the query will retrieve a significant proportion of the relation.
10. Avoid indexing attributes that consist of long character strings.

STEP 4.4 ESTIMATE DISK SPACE REQUIREMENTS

- To estimate the amount of disk space that will be required by the database.

STEP 5 DESIGN USER VIEWS

- To design the user views that were identified during the Requirements Collection and Analysis stage of the database system development lifecycle.

STEP 6 DESIGN SECURITY MEASURES

- To design the security measures for the database as specified by the users.