

Introduction to Object Relational Databases

Chapter 28

Objectives

- How relational model has been extended to support advanced database applications.
- Features proposed in third-generation database system manifestos from CADF and Darwen/Date.
- Extensions to relational data model in Postgres.
- Object-oriented features in SQL:2003.
- Extensions to QP to support advanced queries.
- Object-oriented extensions to Oracle.
- How OODBMSs and ORDBMSs compare in terms of data modeling, data access, and data sharing.

COMP 302

Valentina Tamma

Market Share

- RDBMSs currently dominant database technology with estimated sales \$6 - \$10 billion per year (\$25 billion with tools sales included).
- OODBMS market still small, but still finds new applications areas such as Web.
- Some analysts expect OODBMS market to grow at a faster rate than total database market, but unlikely to overtake relational systems.

COMP 302

Valentina Tamma

ORDBMSs

- Vendors of RDBMSs conscious of threat and promise of OODBMS.
- Agree that RDBMSs not currently suited to advanced database applications, and added functionality is required.
- Reject claim that extended RDBMSs will not provide sufficient functionality or will be too slow to cope adequately with new complexity.
- Can remedy shortcomings of relational model by extending model with OO features.

COMP 302

Valentina Tamma

ORDBMSs - Features

- OO features being added include:
 - user-extensible types,
 - encapsulation,
 - inheritance,
 - polymorphism,
 - dynamic binding of methods,
 - complex objects including non-1NF objects,
 - object identity.

COMP 302

Valentina Tamma

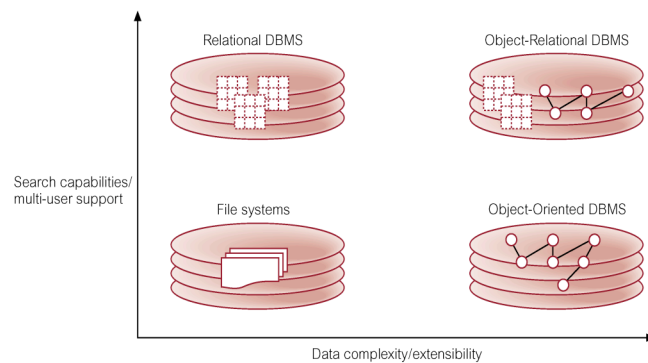
ORDBMSs - Features

- However, no single extended relational model.
- All models:
 - share basic relational tables and query language,
 - all have some concept of 'object',
 - some can store methods (or procedures or triggers).
- Some analysts predict ORDBMS will have 50% larger share of market than RDBMS.

COMP 302

Valentina Tamma

Stonebraker's View



COMP 302

Valentina Tamma

Advantages of ORDBMSs

- Resolves many of known weaknesses of RDBMS.
- Reuse and sharing:
 - reuse comes from ability to extend server to perform standard functionality centrally;
 - gives rise to increased productivity both for developer and end-user.
- Preserves significant body of knowledge and experience gone into developing relational applications.

COMP 302

Valentina Tamma

Disadvantages of ORDBMSs

- Complexity.
- Increased costs.
- Proponents of relational approach believe simplicity and purity of relational model are lost.
- Some believe RDBMS is being extended for what will be a minority of applications.
- OO purists not attracted by extensions either.
- SQL now extremely complex.

COMP 302

Valentina Tamma

CADF Manifesto

- A 3rd generation DBMS must have a rich type system.
- Inheritance is a good idea.
- Functions, including database procedures and methods and encapsulation are a good idea.
- Unique identifiers for records should be assigned by the DBMS only if a user-defined primary key is not available.

COMP 302

Valentina Tamma

CADF Manifesto

- Rules (triggers, constraints) will become a major feature in future. They should not be associated with a specific function or collection.
- Essentially all programmatic access to a database should be through a non-procedural, high-level access language.
- There should be at least two ways to specify collections, one using enumeration of members and one using query language.

COMP 302

Valentina Tamma

CADF Manifesto

- Updateable views are essential.
- Performance indicators have almost nothing to do with data models and must not appear in them.
- Third generation DBMSs must be accessible from multiple high-level languages.
- Persistent forms of a high-level language, for variety of high-level languages, is a good idea. Supported on top of single DBMS by compiler extensions and complex run-time system.

COMP 302

Valentina Tamma

CADF Manifesto

- For better or worse, SQL is “intergalactic dataspeak”.
- Queries and their resulting answers should be the lowest level of communication between a client and a server.

COMP 302

Valentina Tamma

Third Manifesto

- Darwen/Date defend RDM in Third Manifesto.
- Acknowledged that certain OO features desirable, but believe features are orthogonal to RDM.
- Thus, RDM needs ‘no extension, no correction, no subsumption, and, above all, no perversion’.
- However, SQL is *unequivocally rejected* as a *perversion* of model.
- Instead a language called D is proposed.

COMP 302

Valentina Tamma

Third Manifesto

- Primary object is domain - a named set of encapsulated values, of arbitrary complexity, equivalent to data type or object class.
- Domain values referred to as scalars, manipulated only by means of operators defined for domain.
- Both single and multiple inheritance on domains proposed.
- Nested transactions should be supported.

COMP 302

Valentina Tamma

Postgres

- Postgres (‘Post Ingres’) is research DBMS designed to be potential successor to INGRES.
- Some of the objectives of project were to:
 - Provide better support for complex objects.
 - Provide user extensibility for data types, operators, and access methods.
 - Provide active database facilities (alerts and triggers) and inferencing support.
 - Make as few changes as possible (preferably none) to the relational model.

COMP 302

Valentina Tamma

Postgres

- Postgres extended RDM to include:
 - Abstract Data Types,
 - Data of type 'procedure',
 - Rules.
- Supported OO constructs such as aggregation, generalization, complex objects with shared subobjects, and attributes that reference tuples in other relations.

COMP 302

Valentina Tamma

SQL:2003 - New OO Features

- Type constructors for row types and reference types.
- User-defined types (distinct types and structured types) that can participate in supertype/subtype relationships.
- User-defined procedures, functions, methods, and operators.
- Type constructors for collection types (arrays, sets, lists, and multisets).
- Support for large objects – BLOBs and CLOBs.
- Recursion.

COMP 302

Valentina Tamma

Row Types

- Sequence of field name/data type pairs that provides data type to represent types of rows in tables.
- Allows complete rows to be:
 - stored in variables,
 - passed as arguments to routines,
 - returned as return values from function calls.
- Also allows column of table to contain row values.

COMP 302

Valentina Tamma

Use of Row Type

```
CREATE TABLE Branch (branchNo CHAR(4),
    address ROW(street VARCHAR(25),
        city VARCHAR(15),
        postcode ROW(cityIdentifier VARCHAR(4),
            subPart VARCHAR(4))));

INSERT INTO Branch
VALUES ('B005', ('22 Deer Rd', 'London',
    ROW('SW1', '4EH')));
```

COMP 302

Valentina Tamma

User-Defined Types (UDTs)

- SQL:2003 allows definition of UDTs.
- May be used in same way as built-in types.
- Subdivided into two categories: distinct types and structured types.
- Distinct type allows differentiation between same underlying base types:

```
CREATE TYPE OwnerNoType AS VARCHAR(5) FINAL;  
CREATE TYPE StaffNoType AS VARCHAR(5) FINAL;
```

COMP 302

Valentina Tamma

User-Defined Types (UDTs)

- Would get error if attempt to treat instance of one type as instance of other type.
- Not same as SQL domains, which constrains set of valid values that can be stored.
- Generally, UDT definition consists of one or more attribute definitions.
- Definition also consists of routine declarations (operator declarations deferred).
- Can also define equality and ordering relationships using CREATE ORDERING FOR.

COMP 302

Valentina Tamma

UDTs – Encapsulation and get/set functions

- Value of an attribute can be accessed using common dot notation:

```
p.fName      p.fName = 'A. Smith'
```

- SQL encapsulates each attribute through an observer (get) and a mutator (set) function.
- These functions can be redefined by user in UDT definition.

```
FUNCTION fName(p PType) RETURNS VARCHAR(15)  
RETURN p.fName;
```

COMP 302

Valentina Tamma

UDTs – Constructors and NEW expression

- A (public) constructor function is automatically defined to create new instances of type:

```
SET p = NEW PersonType;
```
- The constructor function has same name as type, takes 0 arguments, and returns a new instance with attributes set to their default values.
- User-defined constructor methods can be provided to initialize new instances. Must have same name as UDT but different parameters to public constructor.

COMP 302

Valentina Tamma

UDTs - Example Constructor Method

```
CREATE CONSTRUCTOR METHOD PersonType (  
    fN VARCHAR(15), IN VARCHAR(15), sx CHAR)  
RETURNS PersonType  
BEGIN  
    SET SELF.fName = fN;  
    SET SELF.IName = IN;  
    SET SELF.sex = sx;  
    RETURN SELF;  
END;  
  
SET p = NEW PersonType('John', 'White' 'M');
```

COMP 302

Valentina Tamma

Definition of new UDT

```
CREATE TYPE PersonType AS (  
    dateOfBirth DATE,  
    fName VARCHAR(15) NOT NULL,  
    IName VARCHAR(15) NOT NULL,  
    sex CHAR)  
INSTANTIABLE  
NOT FINAL  
REF IS SYSTEM GENERATED  
INSTANCE METHOD age() RETURNS INTEGER;  
INSTANCE METHOD age(DOB DATE) RETURNS PersonType;
```

COMP 302

Valentina Tamma

Definition of new UDT

```
CREATE INSTANCE METHOD age ()  
RETURNS INTEGER FOR PersonType  
BEGIN  
    RETURN /* set age from SELF.dateOfBirth*/  
END;  
  
CREATE INSTANCE METHOD age(DOB DATE)  
RETURNS PersonType FOR PersonType  
BEGIN  
    SELF.dateOfBirth = /* set dateOfBirth from DOB*/  
    RETURN SELF;  
END;
```

COMP 302

Valentina Tamma

Subtypes and Supertypes

- UDTs can participate in subtype/supertype hierarchy using UNDER clause.
- Multiple inheritance is not supported.
- Subtype inherits all the attributes and behavior of its supertypes.
- Can define additional attributes and methods and can override inherited methods.
- Concept of substitutability supported: whenever instance of supertype expected instance of subtype can be used in its place.

COMP 302

Valentina Tamma

Creation of Subtype

```
CREATE TYPE StaffType UNDER PersonType AS (  
  staffNo    VARCHAR(5),  
  position   VARCHAR(10) DEFAULT 'Assistant',  
  salary     DECIMAL(7, 2),  
  branchNo   CHAR(4))  
INSTANTIABLE  
NOT FINAL  
INSTANCE METHOD isManager() RETURNS BOOLEAN;
```

COMP 302

Valentina Tamma

Creation of Subtype

```
CREATE INSTANCE METHOD isManager ()  
  RETURNS BOOLEAN FOR StaffType  
BEGIN  
  IF SELF.position = 'Manager' THEN  
    RETURN TRUE;  
  ELSE  
    RETURN FALSE;  
  END IF  
END)
```

COMP 302

Valentina Tamma

User-Defined Routines (UDRs)

- UDRs define methods for manipulating data.
- UDRs may be defined as part of a UDT or separately as part of a schema.
- An SQL-invoked routine may be a procedure, function, or method.
- May be externally provided in standard programming language or defined completely in SQL.

COMP 302

Valentina Tamma

User-Defined Routines (UDRs)

- An SQL-invoked procedure is invoked from SQL CALL statement.
- May have zero or more parameters, each of which may be IN, OUT, or INOUT, and a body if defined fully within SQL.
- An SQL-invoked function returns a value.
- Any specified parameters must be input parameters with one designated as result parameter (using RESULT keyword).

COMP 302

Valentina Tamma

User-Defined Routines (UDRs)

- An SQL-invoked procedure is invoked from SQL CALL statement.
- May have zero or more parameters, each of which may be IN, OUT, or INOUT, and a body if defined fully within SQL.
- An SQL-invoked function returns a value.
- Any specified parameters must be input parameters with one designated as result parameter (using RESULT keyword).

COMP 302

Valentina Tamma

User-Defined Routines (UDRs)

- SQL-invoked method is similar to a function but:
 - method is associated with a single UDT;
 - signature of every method of a UDT must be specified in UDT and definition of method must specify UDT.
- Three types of methods:
 - constructor methods, invoked using NEW;
 - instance methods, invoked using dot notation or using generalized invocation format; e.g. p.fName or (p AS StaffType).fName();
 - static methods (analogous to class methods), invoked using ::; e.g. StaffType::totalStaff().

COMP 302

Valentina Tamma

User-Defined Routines (UDRs)

- External routine defined by specifying an external clause that identifies 'compiled code' in operating system's file storage.
- ORDBMS will provide method to dynamically link this object file into the DBMS so that it can be invoked when required.
- Procedure for this is outside bounds of SQL standard and is left as implementation-defined.

COMP 302

Valentina Tamma

Polymorphism

- Routine names may be overloaded, provided:
 - no two functions in same schema have same signature;
 - no two procedures in same schema have same name and number of parameters.
- Overriding applies only to methods and only based on runtime value of SELF argument.
- SQL:2003 uses generalized object model, so types of all arguments considered when deciding which routine to invoke (left to right).
- Precedence lists used to determine closest match.

COMP 302

Valentina Tamma

Reference Types and Object Identity

- In SQL:2003, reference types can be used to define relationships between row types and uniquely identify a row within a table.
- Reference type value can be stored in one table and used as a direct reference to a specific row in some base table defined to be of this type (similar to pointer type in 'C'/C++).
- In this way, reference type provides similar functionality as OID of OODBMSs.

COMP 302

Valentina Tamma

Reference Types and Object Identity

- Thus, references allow a row to be shared among multiple tables, and enable users to replace complex join definitions in queries with much simpler path expressions.
- References also give optimizer alternative way to navigate data instead of using value-based joins.
- REF IS SYSTEM GENERATED in CREATE TYPE indicates that actual values of associated REF type are provided by the system.

COMP 302

Valentina Tamma

Table Creation based on UDT

```
CREATE TABLE Person (  
  info      PersonType,  
  CONSTRAINT DOB_Check  
    CHECK(dateOfBirth > DATE'1900-01-01');
```

• or

```
CREATE TABLE Person OF PersonType (  
  dateOfBirth WITH OPTIONS  
  CONSTRAINT DOB_Check  
    CHECK(dateOfBirth > DATE'1900-01-01')  
  REF IS personID SYSTEM GENERATED);
```

COMP 302

Valentina Tamma

Using Reference Type to Define a Relationship

```
CREATE TABLE PropertyForRent (  
  propertyNo  PropertyNumber  NOT NULL,  
  street      Street          NOT NULL,  
  ....  
  staffID     REF(StaffType)  SCOPE Staff  
  REFERENCES ARE CHECKED  
  ON DELETE CASCADE,  
  PRIMARY KEY (propertyNo));
```

COMP 302

Valentina Tamma

Subtables and Supertables

- No mechanism to store all instances of given UDT, unless user explicitly creates a single table in which all instances are stored.
- Thus, in SQL:2003 may not be possible to apply an SQL query to all instances of a given UDT.
- Can use table inheritance, which allows table to be created that inherits all the attributes of one or more existing tables using UNDER clause.
- Subtable/supertable independent from UDT inheritance facility.

COMP 302

Valentina Tamma

Creation of Subtable

```
CREATE TABLE Staff OF StaffType UNDER Person;
```

- Each row of supertable Person can correspond to at most one row in Staff.
- Each row in Staff must have exactly one corresponding row in Person.
- Containment used: row of subtable 'contained' in one of its supertables.

COMP 302

Valentina Tamma

Retrieve Specific Column/Rows

Find the names of all Managers.

```
SELECT s.IName  
FROM Staff s  
WHERE s.position = 'Manager';
```

- Uses implicitly defined observer function position.

COMP 302

Valentina Tamma

Invoke User-Defined Function

Find the names and ages of all Managers.

```
SELECT s.IName, s.age  
FROM Staff s  
WHERE s.isManager;
```

- Uses user-defined function isManager as a predicate of the WHERE clause (returns TRUE if member of staff is a manager).
- Also uses inherited virtual observer function age.

COMP 302

Valentina Tamma

Use of ONLY

Find names of all people over 65.

```
SELECT p.lName, p.fName
FROM Person p
WHERE p.age > 65;
```

- This will list out not only records explicitly inserted into Person table, but also records inserted directly/indirect into subtables of Person.

COMP 302

Valentina Tamma

Use of ONLY

- Can restrict access to specific instances of Person table, excluding any subtables, using ONLY.

```
SELECT p.lName, p.fName
FROM ONLY (Person) p
WHERE p.age > 65;
```

COMP 302

Valentina Tamma

Use of Dereference Operator

Find name of member of staff who manages property PG4.

```
SELECT p.staffID->fName AS fName,
       p.staffID->lName AS lName,
FROM PropertyForRent p
WHERE p.propertyNo = 'PG4';
```

- In SQL2, this query would have required a join or nested subquery.

COMP 302

Valentina Tamma

Use of Dereference Operator

To retrieve the member of staff for property PG4, rather than just the first and last name:

```
SELECT Deref(p.staffID) AS Staff
FROM PropertyForRent p
WHERE p.propertyNo = 'PG4';
```

- Note, reference types by themselves do not provide referential integrity.

COMP 302

Valentina Tamma

Collection Types

- Collections are type constructors used to define collections of other types.
- Used to store multiple values in single column and can result in nested tables.
- SQL:2003 has parameterized ARRAY and MULTiset collection types.
- Later SQL may have parameterized LIST and SET collection types.
- The parameter may be predefined type, UDT, row type, or another collection (but not reference type or UDT containing reference type).

COMP 302

Valentina Tamma

Collection Types

- ARRAY: 1D array with maximum number of elements.
 - LIST: ordered collection that allows duplicates.
 - SET: unordered collection that does not allow duplicates.
 - MULTiset: unordered collection that allows duplicates.
- Similar to those in the ODMG 3.0 standard .

COMP 302

Valentina Tamma

Use of ARRAY Collection

- Branch has up to three telephone numbers:

```
telNo    VARCHAR(13) ARRAY[3]
```

Retrieve first phone number for B003.

```
SELECT telNo[1]
FROM Branch
WHERE branchNo = 'B003';
```

COMP 302

Valentina Tamma

MULTiset

- Unordered collection of elements, all of same type, with duplicates permitted.
- Since multiset is unordered there is no ordinal position to reference individual elements. Unlike arrays, multiset is an unbounded collection.
- Analogous to tables, operators are provided to convert multiset to table (UNNEST) and table to multiset (MULTiset).

COMP 302

Valentina Tamma

Operations on MULTISSET

- SET, removes duplicates from a multiset to produce a set.
- CARDINALITY, returns number of current elements.
- ELEMENT, returns element of a multiset if multiset only has one element (or null if multiset has no elements). Exception raised if multiset has more than one element.

COMP 302

Valentina Tamma

Operations on MULTISSET

- MULTISSET UNION, computes union of two multisets; keywords ALL or DISTINCT can retain duplicates or remove them.
- MULTISSET INTERSECT, computes intersection of two multisets; keyword DISTINCT can remove duplicates; keyword ALL can place in result as many instances of each value as minimum number of instances of that value in either operand.
- MULTISSET EXCEPT, computes difference of two multisets; again, keyword DISTINCT or ALL can be specified.

COMP 302

Valentina Tamma

Aggregate Functions for MULTISSET

- COLLECT, creates multiset from value of the argument in each row of a group;
- FUSION, creates multiset union of a multiset value in all rows of a group;
- INTERSECTION, creates multiset intersection of a multiset value in all rows of a group.

COMP 302

Valentina Tamma

Predicates for use with MULTISSET

- Comparison predicate (equality and inequality only);
- DISTINCT predicate;
- MEMBER predicate;
- SUBMULTISSET predicate, which tests whether one multiset is a submultiset of another;
- IS A SET/IS NOT A SET predicate, which checks whether a multiset is a set.

COMP 302

Valentina Tamma

Use of Collection MULTISSET

- Extend Staff table to contain details of next-of-kin:

nok NameType MULTISSET

Find first and last names of John White's next of kin.

```
SELECT n.fName, n.lName
FROM Staff s, UNNEST(s.nok) AS n(fName, lName)
WHERE s.lName = 'White' AND s.fName = 'John';
```

COMP 302

Valentina Tamma

FUSION and INTERSECTION

propertyNo	viewDates
PA14	MULTISSET['14-May-04', '24-May-04']
PG4	MULTISSET['20-Apr-04', '14-May-04', '26-May-04']
PG36	MULTISSET['28-Apr-04', '14-May-04']
PL94	Null

```
SELECT FUSION(viewDates) AS viewDateFusion,
INTERSECTION(viewDates) AS viewDateIntersection
FROM PropertyViewDates;
```

COMP 302

Valentina Tamma

FUSION and INTERSECTION

viewDateFusion	viewDateIntersection
MULTISSET['14-May-04', '14-May-04', '14-May-04', '24-May-04', '20-Apr-04', '26-May-04', '28-Apr-04']	MULTISSET['14-May-04']

COMP 302

Valentina Tamma

Typed Views

```
CREATE VIEW FemaleView OF
PersonType (REF IS personID DERIVED)
AS SELECT fName, lName
FROM ONLY (Person)
WHERE sex = 'F';
CREATE VIEW FemaleStaff3View OF
StaffType UNDER FemaleView
AS SELECT fName, lName, staffNo, position
FROM ONLY (Staff)
WHERE branchNo = 'B003';
```

COMP 302

Valentina Tamma

Persistent Stored Modules (SQL/PSM)

- SQL:2003 has some new statement types to make it computationally complete.
- Behavior (methods) can be stored and executed from within database as SQL statements.
- Can group statements into a compound statement (block), with its own local variables.

COMP 302

Valentina Tamma

Persistent Stored Modules (SQL/PSM)

- Some of the new statements are:
 - An assignment statement.
 - An IF ... THEN ... ELSE ... END IF statement.
 - CASE statement.
 - A set of statements for iteration: FOR, WHILE, and REPEAT.
 - A CALL statement to invoke procedures and a RETURN statement.

COMP 302

Valentina Tamma

SQL/PSM - Condition Handling

- SQL/PSM includes condition handling to handle exceptions and completion conditions.
- First define handler by specifying its type, exception and completion conditions it can resolve, and action it takes to do so (an SQL procedure statement).
- Provides ability to explicitly signal exception and completion conditions, using SIGNAL/ RESIGNAL statement.

COMP 302

Valentina Tamma

Triggers

- An SQL (compound) statement executed automatically by DBMS as side effect of a modification to named table.
- Use of triggers include:
 - Validating input data and maintaining complex integrity constraints that otherwise would be difficult/impossible.
 - Supporting alerts.
 - Maintaining audit information.
 - Supporting replication.

COMP 302

Valentina Tamma

Triggers

```
CREATE TRIGGER TriggerName
BEFORE | AFTER <triggerEvent>
    ON <TableName>
[REFERENCING <oldOrNewValuesAliasList>]
[FOR EACH {ROW | STATEMENT}]
[ WHEN (triggerCondition) ]
<triggerBody>
```

COMP 302

Valentina Tamma

Triggers

- BEFORE trigger fired before and AFTER trigger is fired after associated event occurs.
- Triggered action is SQL procedure statement, which can be executed in one of two ways:
 - For each row (FOR EACH ROW) affected by the event. This is called a row-level trigger.
 - Only once for entire event (FOR EACH STATEMENT), which is default. This is called a statement-level trigger.

COMP 302

Valentina Tamma

Triggers

- As more than one trigger can be defined on a table, order of firing is important. The following order is observed:
 - (1) Execution of any BEFORE triggers on table.
 - (2) For each row affected by the statement:
 - Execute any BEFORE row-level trigger.
 - Execute the statement itself.
 - Apply any referential constraints.
 - Execute any AFTER row-level trigger.
 - (3) Execute any AFTER trigger on table.

COMP 302

Valentina Tamma

Use of AFTER Trigger

```
CREATE TRIGGER InsertMailshotTable
AFTER INSERT ON PropertyForRent
REFERENCING NEW ROW AS pfr
BEGIN
    INSERT INTO Mailshot VALUES
    (SELECT c.fName, c.lName, c.maxRent,
    pfr.propertyNo, pfr.street, pfr.city, pfr.postcode,
    pfr.type, pfr.rooms, pfr.rent
    FROM Client c
    WHERE c.branchNo = pfr.branchNo AND
    (c.prefType=pfr.type AND c.maxRent <= pfr.rent) )
END;
```

COMP 302

Valentina Tamma

Use of AFTER Trigger with Condition

```
CREATE TRIGGER UpdateMailshotTable
AFTER UPDATE OF rent ON PropertyForRent
REFERENCING NEW ROW AS pfr
FOR EACH ROW
BEGIN
    DELETE FROM Mailshot
    WHERE maxRent > pfr.rent;
    UPDATE Mailshot SET rent = pfr.rent
    WHERE propertyNo = pfr.propertyNo;
END;
```

COMP 302

Valentina Tamma

Triggers - Advantages and Disadvantages

- Major advantage - standard functions can be stored within database and enforced consistently.
- This can dramatically reduce complexity of applications.
- However, there can be some disadvantages:
 - Complexity.
 - Hidden functionality.
 - Performance overhead.

COMP 302

Valentina Tamma

Large Objects

- A table field that holds large amount of data.
- Three different types:
 - Binary Large Object (BLOB).
 - Character LOB (CLOB) and National CLOB.
- SQL:2003 LOB slightly different from original type of BLOB that appears in many current DBMSs, where BLOB is non-interpreted byte stream.
- In SQL:2003, LOB does allow some operations to be carried out in DBMS server.

COMP 302

Valentina Tamma

Use of CLOB and BLOB

Extend Staff table to hold a resume and picture for the staff member.

```
ALTER TABLE Staff
    ADD COLUMN resume CLOB(50K);
ALTER TABLE Staff
    ADD COLUMN picture BLOB(12M);
```

COMP 302

Valentina Tamma

Recursion

- Linear recursion is new SQL operation.

```
WITH RECURSIVE
AllManagers(staffNo, managerStaffNo) AS
(SELECT staffNo, managerStaffNo
FROM Staff
UNION
SELECT in.staffNo, out.managerStaffNo
FROM AllManagers in, Staff out
WHERE in.managerStaffNo = out.staffNo);
SELECT * FROM AllManagers
ORDER BY staffNo, managerStaffNo;
```

COMP 302

Valentina Tamma

Recursion

- Application may require data to be inserted into result table in certain order:
 - depth-first, where each 'parent' or 'containing' item appears in result before items that it contains, as well as before its 'siblings';
 - breadth-first, where items follow their 'siblings' without following siblings' children.

COMP 302

Valentina Tamma

Recursion

- If data can be recursive (not just structure), an infinite loop can occur unless cycle can be detected.
- CYCLE clause instructs SQL to record a specified value to indicate that a new row has already been added to result table.
- Whenever new row is found, SQL checks that row has not been added previously by determining whether row has been marked with specified value. If it has, then SQL assumes cycle has been encountered and stops searching for further result rows.

```
CYCLE staffNo, managerStaffNo
SET cycleMark TO 'Y' DEFAULT 'N'
USING cyclePath
```

COMP 302

Valentina Tamma

Query Processing and Optimization

- SQL:2003 does not address some areas of extensibility, such as mechanisms for:
 - defining new index structures;
 - giving query optimizer cost information about UDFs will vary among products.
- Lack of standard way to integrate software with multiple ORDBMSs shows need for standards beyond focus of SQL:2003.

COMP 302

Valentina Tamma

Use of UDFs Revisited

List flats that are for rent at branch B003.

```
CREATE FUNCTION flatTypes()
  RETURNS SET(PropertyForRent)
  SELECT * FROM PropertyForRent
  WHERE type = 'Flat';

SELECT propertyNo, street, city, postcode
FROM TABLE (flatTypes())
WHERE branchNo = 'B003';
```

COMP 302

Valentina Tamma

Use of UDFs Revisited

• QP should 'flatten' this query:

```
(1) SELECT propertyNo, street, city, postcode
     FROM TABLE (SELECT * FROM PropertyForRent
                  WHERE type = 'Flat')
     WHERE branchNo = 'B003';

(2) SELECT propertyNo, street, city, postcode
     FROM PropertyForRent
     WHERE type = 'Flat' AND branchNo = 'B003';
```

COMP 302

Valentina Tamma

Query Processing and Optimization

- ORDBMS could flatten query here because UDF had been implemented in SQL.
- If UDF had been defined as an external function, then need to be able to provide information to optimize query execution.
- May be difficult for user to provide these figures.
- Could ask ORDBMS to derive these figures based on experimentation.

COMP 302

Valentina Tamma

Different QP Heuristics

Find all detached properties in Glasgow that are within 2 miles of a primary school and are managed by Ann Beech.

```
SELECT *
FROM PropertyForRent p, staff s
WHERE p.staffNo = s.staffNo AND
      p.nearPrimarySchool(p.postcode) < 2.0 AND
      p.city = 'Glasgow' AND
      s.fName = 'Ann' AND s.lName = 'Beech';
```

COMP 302

Valentina Tamma

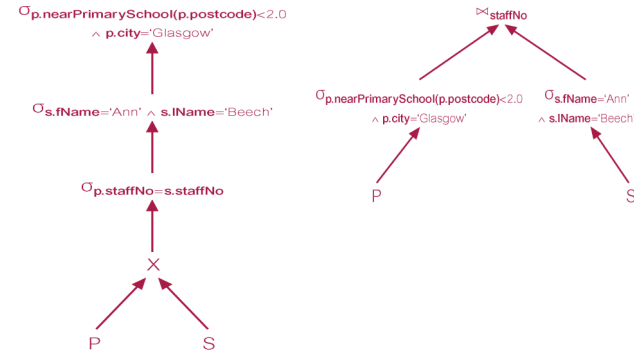
Different QP Heuristics

- Generally, would push selection down past CP and transform CP/selection into join.
- This may not be best strategy here.
- If UDF has large amount of processing, better to perform selection on Staff first and then perform join on staffNo before calling UDF.
- Use commutativity of join to rearrange leaf nodes, so more restrictive selection performed first.
- Also evaluate selection "city = 'Glasgow'" before UDF.

COMP 302

Valentina Tamma

Different QP Heuristics



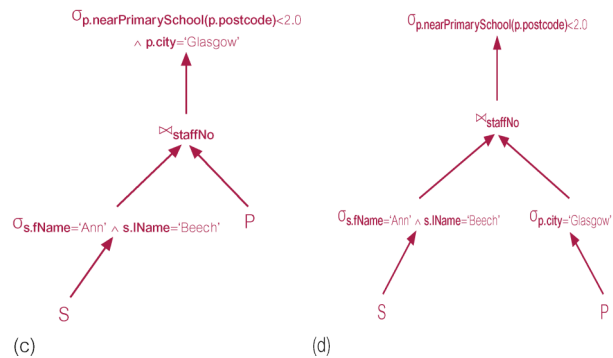
(a)

COMP 302

(b)

Valentina Tamma

Different QP Heuristics



(c)

(d)

COMP 302

Valentina Tamma

New Index Types

- ORDBMS can compute and index result of a UDF that returns scalar data.
- RDBMSs use B-tree indexes to speed access to scalar data.
- However, B-tree is a 1D access method, inappropriate for multidimensional access.
- Specialized index structures are required for efficient access to data.

COMP 302

Valentina Tamma

New Index Types

- Some ORDBMSs now support additional indexes:
 - Generic B-trees that allow B-trees to be built on any data type, not just alphanumeric.
 - Quad trees.
 - K-D-B trees.
 - R-trees for fast access to 2D/3D data.
 - Grid files.
 - D-Trees for text support.

COMP 302

Valentina Tamma

New Index Types

- A mechanism to plug in any user-defined index structure provides greatest flexibility.
- Generalized Search Tree (GiST) is template index structure based on B-trees, which accommodates many tree-based index structures with minimal coding.

COMP 302

Valentina Tamma

Object-Oriented Extensions in Oracle

- Many of the object-oriented features that appear in new SQL:2003 standard appear in Oracle in one form or another.
- Oracle supports two user-defined data types:
 - object types;
 - collection types.

COMP 302

Valentina Tamma

Object Types in Oracle

- An object type is a schema object that has a name, a set of attributes based on the Oracle built-in data types or possibly other object types, and a set of methods.

```
CREATE TYPE AddressType AS OBJECT (  
    street          VARCHAR2(25),  
    city            VARCHAR2(15),  
    postcode        VARCHAR2(8));
```

COMP 302

Valentina Tamma

Object-Oriented Extensions in Oracle

```
CREATE TYPE StaffType AS OBJECT (  
    staffNo VARCHAR2(5),  
    fName VARCHAR2(15),  
    ....  
    MAP MEMBER FUNCTION age  
        RETURN INTEGER,  
    PRAGMA RESTRICT_REFERENCES(  
        age, WNDS, WNPS, RNPS));
```

COMP 302

Valentina Tamma

Object-Oriented Extensions in Oracle

```
CREATE TYPE BranchType AS OBJECT (  
    branchNo VARCHAR2(4),  
    address AddressType,  
    MAP MEMBER FUNCTION getbranchNo  
        RETURN VARCHAR2(4),  
    PRAGMA RESTRICT_REFERENCES(  
        getbranchNo, WNDS, WNPS, RNDS, RNPS));
```

COMP 302

Valentina Tamma

Object Types in Oracle

- Pragma clause is a compiler directive that denies member functions read/write access to database tables and/or package variables.
- Can now create a Branch (Object) table:

```
CREATE TABLE Branch OF BranchType  
    (branchNo PRIMARY KEY);
```

COMP 302

Valentina Tamma

Methods in Oracle

- Methods of an object type are classified as member, static, and comparison.
- Member method is a function/procedure that always has implicit SELF parameter as first parameter (whose type is containing object type).
 - Useful as observer and mutator functions.
- Static method is a function/procedure that does not have an implicit SELF parameter.
 - Useful for specifying user-defined constructors or cast methods and may be invoked by qualifying method with the type name, as in typename.method().

COMP 302

Valentina Tamma

Methods in Oracle

- Comparison method used for comparing instances of objects.
- Oracle provides two ways to define an order relationship among objects of a given type:
 - a map method uses Oracle's ability to compare built-in types.
 - an order method uses its own internal logic to compare two objects of a given object type. It returns a value that encodes the order relationship. For example, may return -1 if first is smaller, 0 if they are equal, and 1 if first is larger.

COMP 302

Valentina Tamma

Methods in Oracle

- Methods can be implemented in PL/SQL, Java, and 'C'.
- Overloading is supported provided their formal parameters differ in number, order, or data type.

COMP 302

Valentina Tamma

Object Identifiers

- Every row object in an object table has associated logical OID, which uniquely identifies the row.
- The OID column is hidden from users and there is no access to its internal structure.
- Oracle requires every row object to have a unique OID, which may be specified to come from the row object's PK or to be system-generated.

```
CREATE TABLE Branch OF BranchType
    (branchNo PRIMARY KEY)
    OBJECT IDENTIFIER PRIMARY KEY;
```

COMP 302

Valentina Tamma

REF Data Type

- Oracle provides a built-in data type called REF to encapsulate references to row objects of a specified object type.
- In effect, a REF is used to model an association between two row objects.
- A REF can be used to examine or update the object it refers to and to obtain a copy of the object it refers to.
- Only changes that can be made to a REF are to replace its contents with a reference to a different object of same object type or to assign it a null value.

COMP 302

Valentina Tamma

REF Data Type

```
CREATE TYPE BranchType AS OBJECT (  
  branchNo      VARCHAR2(4),  
  address       AddressType,  
  manager       REF StaffType,  
  MAP MEMBER FUNCTION  
    getbranchNo RETURN VARCHAR2(4),  
  PRAGMA RESTRICT_REFERENCES(  
    getbranchNo, WNDS, WNPS, RNDS, RNPS));
```

COMP 302

Valentina Tamma

Collection Types

- Oracle supports two collection types: array types and table types.
- An array is an ordered set of data elements, all of same data type.
- Each element has an *index*, a number corresponding to the element's position in the array.
- An array can have a fixed or variable size, although in latter case maximum size must be specified when array type is declared.

COMP 302

Valentina Tamma

Nested Tables

- An unordered set of data elements, all of same data type.
- It has a single column of a built-in type or an object type.
- If column is an object type, table can also be viewed as a multi-column table, with a column for each attribute of the object type.

COMP 302

Valentina Tamma

Nested Tables

```
CREATE TYPE NextOfKinType AS OBJECT (  
  fName        VARCHAR2(15),  
  lName        VARCHAR2(15),  
  telNo        VARCHAR2(13));
```

```
CREATE TYPE NextOfKinNestedType AS  
  TABLE OF NextOfKinType;
```

- Can now modify StaffType to include this new type:
 nextOfKin NextOfKinNestedType

COMP 302

Valentina Tamma

Nested Tables

- Can now create Staff table:

```
CREATE TABLE Staff OF StaffType (  
  PRIMARY KEY   staffNo)  
  OBJECT IDENTIFIER PRIMARY KEY  
  NESTED TABLE nextOfKin STORE AS NextOfKinStorageTable (  
    (PRIMARY KEY(Nested_Table_Id, IName, telNo))  
    ORGANIZATION INDEX COMPRESS)  
  RETURN AS LOCATOR;
```

COMP 302

Valentina Tamma

Manipulating Object Tables

```
INSERT INTO Staff VALUES ('SG37', 'Ann',  
  'Beech', 'Assistant', 'F', '10-Nov-1960', 12000,  
  NextOfKinNestedType());  
  
INSERT INTO TABLE (SELECT s.nextOfKin  
  FROM Staff s  
  WHERE s.staffNo = 'SG5')  
VALUES ('John', 'Brand', '0141-848-2000');
```

COMP 302

Valentina Tamma

Manipulating Object Tables

- Can now insert object into Branch table:

```
INSERT INTO Branch  
  SELECT 'B003', AddressType('163 Main St',  
    'Glasgow', 'G11 9QX'), REF(s),  
    TelNoArrayType('0141-339-2178',  
      '0141-339-4439')  
  
  FROM Staff s  
  WHERE s.staffNo = 'SG5';
```

COMP 302

Valentina Tamma

Querying Object Tables

```
SELECT b.branchNo  
  FROM Branch b  
  ORDER BY VALUE(b);  
  
SELECT b.branchNo, b.address,  
  Deref(b.manager), b.phoneList  
  FROM Branch b  
  WHERE b.address.city = 'Glasgow'  
  ORDER BY VALUE(b);
```

COMP 302

Valentina Tamma

Object Views

- *Object view* is a virtual object table.
- In Oracle, can create an object view that not only restricts access to some data but also prevents some methods from being invoked, such as a delete method.
- It has also been argued that object views provide a simple migration path from a purely relational-based application to an object-oriented one, thereby allowing companies to experiment with this new technology.

COMP 302

Valentina Tamma

Data Modeling Comparison of ORDBMS and OODBMS

Table 28.2 Data modeling comparison of ORDBMS and OODBMS.

Feature	ORDBMS	OODBMS
Object identity (OID)	Supported through REF type	Supported
Encapsulation	Supported through UDTs	Supported but broken for queries
Inheritance	Supported (separate hierarchies for UDTs and tables)	Supported
Polymorphism	Supported (UDF invocation based on the generic function)	Supported as in an object-oriented programming model language
Complex objects	Supported through UDTs	Supported
Relationships	Strong support with user-defined referential integrity constraints	Supported (for example, using class libraries)

COMP 302

Valentina Tamma

Data Access Comparison of ORDBMS and OODBMS

Table 28.3 Data access comparison of ORDBMS and OODBMS.

Feature	ORDBMS	OODBMS
Creating and accessing persistent data	Supported but not transparent	Supported but degree of transparency differs between products
<i>Ad hoc</i> query facility	Strong support	Supported through ODMG 3.0
Navigation	Supported by REF type	Strong support
Integrity constraints	Strong support	No support
Object server/page server	Object server	Either
Schema evolution	Limited support	Supported but degree of support differs between products

COMP 302

Valentina Tamma

Data Sharing Comparison of ORDBMS and OODBMS

Table 28.4 Data sharing comparison of ORDBMS and OODBMS.

Feature	ORDBMS	OODBMS
ACID transactions	Strong support	Supported
Recovery	Strong support	Supported but degree of support differs between products
Advanced transaction models	No support	Supported but degree of support differs between products
Security, integrity, and views	Strong support	Limited support

COMP 302

Valentina Tamma