Advanced Database Systems SET09107

Object-Relational Databases

Structured Types

Nested Relations

- Motivation:
  - Permits non-atomic domains
    - ➢ (atomic ≡ indivisible)
  - Example of non-atomic domain: set of integers, or set of tuples, date = day, month, year
  - Allows more intuitive modelling for applications with complex data

Nested Relations -- Cont'd

- Intuitive definition:
  - Allows relations wherever we allow atomic (scalar) values
    - ➢ relations within relations
  - Retains mathematical foundation of relational model
  - Violates first normal form.

Example of a Nested Relation

- Example: library information system
- Each book has
  - title,
  - a set of authors,
  - publisher, and
  - a set of keywords

Example – Cont'd

- Non-1NF relation books

Title	Author-set	Publisher	Keyword-set
		(name,branch)	
Database Systems	{Connolly, Begg}	(Addison Wesley, New York)	Relational database, normalisation}
Database Systems Concept	{Silberschatz, Korth, Sudarshan}	(McGraw Hill, Singapore)	{Object-Based Databases, Object-Relational}

1NF Version of Nested Relation

- 1NF version of books

Title	Author-set	Publisher-name	Publisher-branch	Keyword-set
Database Systems	Connolly	Addison Wesley	New York	Relational db
Database Systems	Begg	Addison Wesley	New York	Relational db
Database Systems	Connolly	Addison Wesley	New York	normalisation
Database Systems	Begg	Addison Wesley	New York	normalisation
Database System Concepts	Silberschatz	McGraw Hill	Singapore	Object-Based Databases
Database System Concepts	Korth	McGraw Hill	Singapore	Object-Based Databases
Database System Concepts	Sudarshan	McGraw Hill	Singapore	Object-Based Databases
Database System Concepts	Silberschatz	McGraw Hill	Singapore	Object - Relational
Database System Concepts	Korth	McGraw Hill	Singapore	Object - Relational
Database System Concepts	Sudarshan	McGraw Hill	Singapore	Object - Relational

4NF Decomposition of Nested Relation

- Remove awkwardness of flat-books by assuming that the following multi-valued dependencies hold:
  - title author
  - title keyword
  - title pub-name, pub-branch
- Decompose flat-doc into 4NF using the schemas:
  - (title, author)
  - (title, keyword)
  - (title, pub-name, pub-branch)

4NF Decomposition of flat– books

Title	Author
Database Systems	Connolly
Database Systems	Begg
Database system Concepts	Silberschatz
Database system Concepts	Korth
Database system Concepts	Sudarshan

Authors

Title	Keyword
Database Systems	Relational database
Database Systems	normalisation
Database System Concepts	Object-Based Databases
Database System Concepts	Object-Relational

Keywords

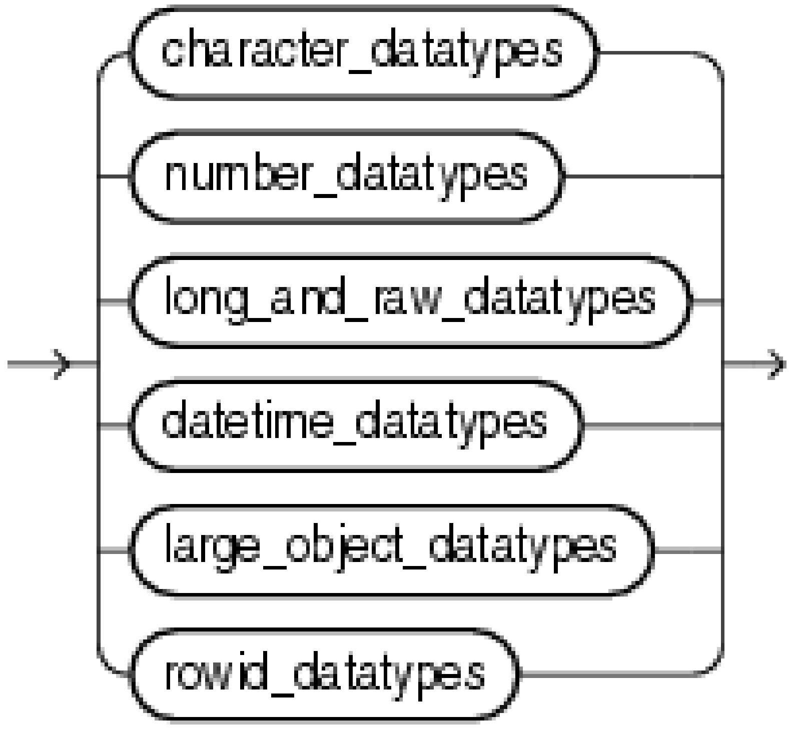
Title	Publisher-name	Publisher-branch
Database Systems	Addison Wesley	New York
Database System Concepts	McGraw Hill	Singapore

Books

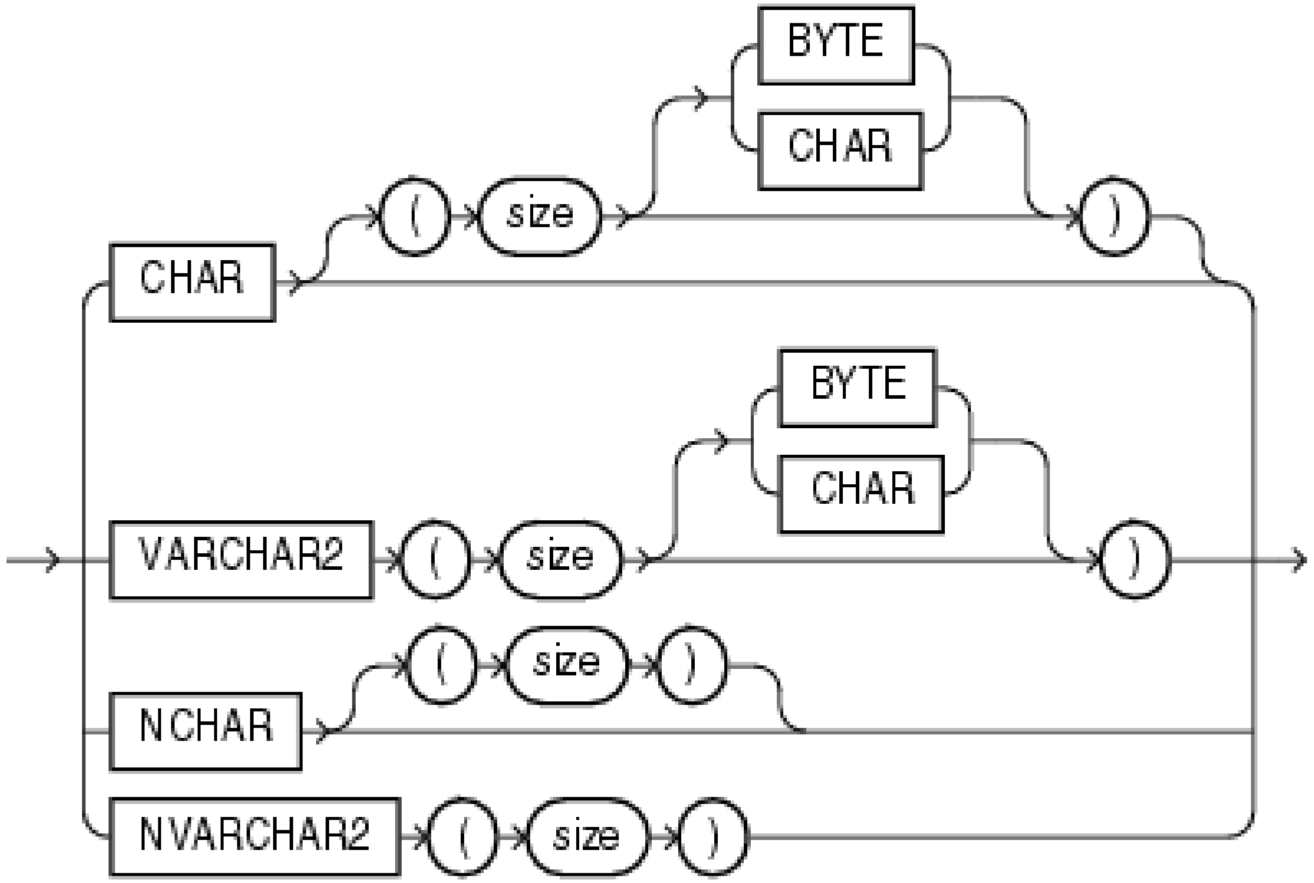
Problems with 4NF Schema

- 4NF design requires users to include joins in their queries.
- 1NF relational view flat-books defined by join of 4NF relations:
  - eliminates the need for users to perform joins,
  - but loses the one-to-one correspondence between tuples and documents.
  - and has a large amount of redundancy
- Nested relations representation is much more natural here.

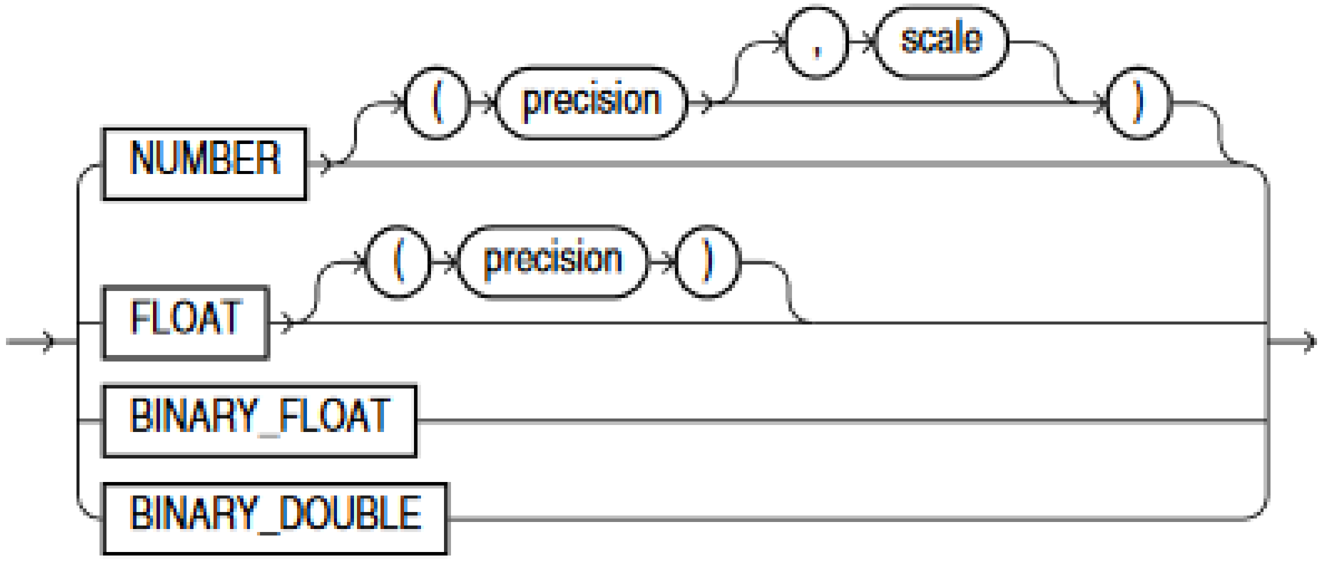
Oracle Built-in data Types



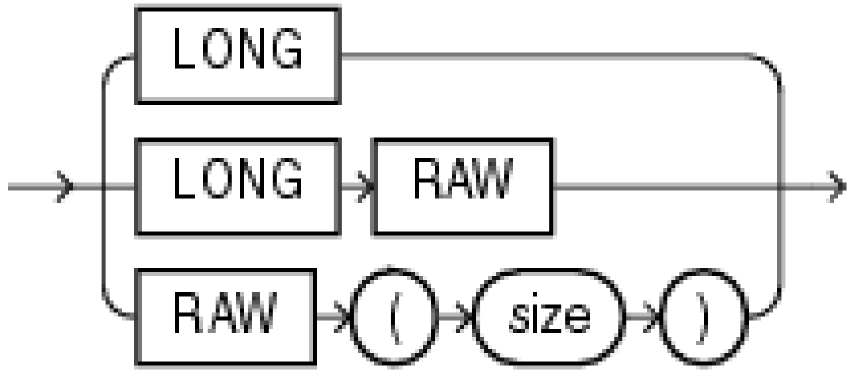
Character data Types



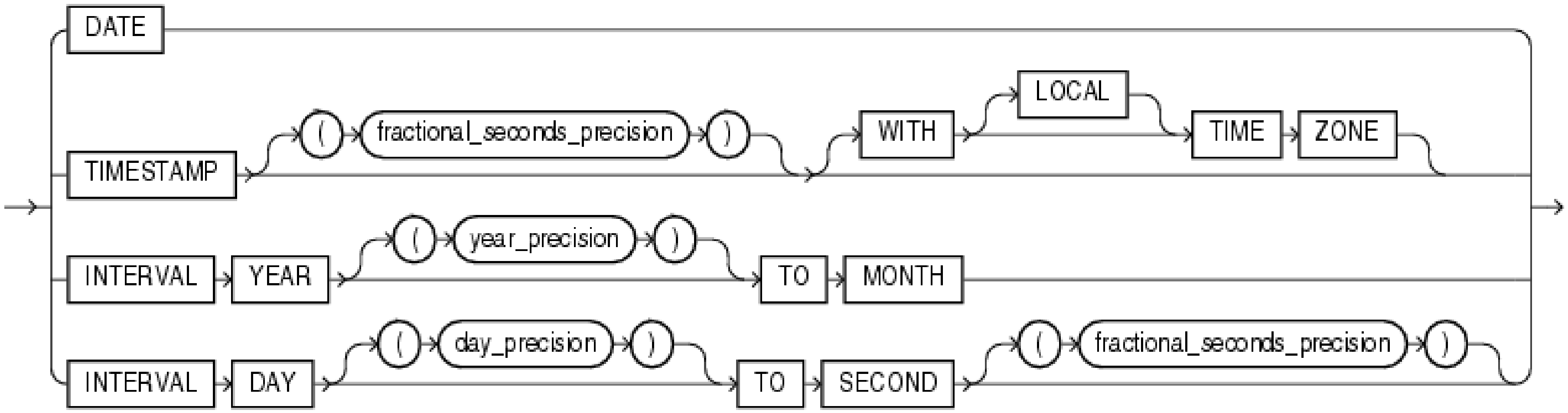
Number data Types



Long/Raw Data Types



Date Time Data Types



VARCHAR vs. VARCHAR2

- Both are used to store variable length character strings
- VARCHAR – maximum bytes of characters 2000.
- VARCHAR2 – maximum bytes of characters 4000
- Currently they behave exactly the same
- Oracle recommends VARCHAR2
- VARCHAR2(20) or VARCHAR2(20 CHAR)

Built-in data Types - Explanation

See

2-1 Datatypes in

Oracle® Database SQL Language Reference 11*g* Release 1

for details

Structured Types

- Structured types can be declared:

```
create type Name as object
      ( firstname varchar2(20),
      surname varchar2(20))
      final
create type Address as object
      (street varchar2(20),
      city varchar2(20),
      postal_code varchar2(8))
      not final
```

  – These are called user-defined types
  – The final specification indicates subtypes are not allowed for this type
  – The not final indicates subtypes are allowed

Types & Composite Attributes

- Use structured types to create composite attributes in a relation
- A table can be created

```
create table people
        ( pname Name,
        paddress  Address,
        dateOfBirth date);
```

- components of a composite attribute can be accessed using a “dot” notation, such as pname.firstname

Types & Tables

- Tables also can be defined as

```
drop type peopleType force;
-- if previously created

create type peopleType as object
        ( pname Name,
        paddress  Address,
        dateOfBirth date)
        not final

create table peopleTable of peopleType;
```

Insert values

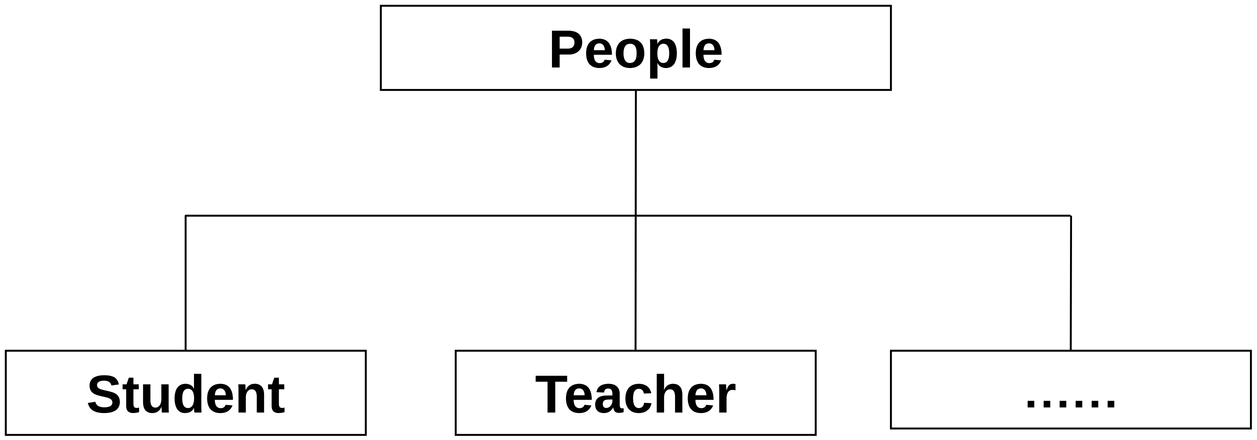
Insert into *peopleTable*
  values
   (Name('John', 'Smith'),
    Address('10 Merchiston', 'Edinburgh', 'EH10 5DT'),
   '21-Feb-89'
   );

Access Component Attributes

select *p.pname.surname, p.paddress.city*
from *peopleTable* p*;*

Subtypes

- Generally related to the **supertype**



Subtypes-- Cont'd

- Subtypes can be created with some extra attributes:

```
create type Student under peopleType
    ( programme varchar2(20),
    school varchar2(20))
    final
create type Teacher under peopleType
    (salary number,
    school varchar2(20))
    final
```

Subtypes-- Cont'd

- Tables and sub-tables can be created as follows:

create table *peopleTable* **of** *peopleType*;

create table *studentTable* **of** *student*;

Subtypes-- Examples

- Insert values to sub-tables:

```
Insert into studentTable
   values
     (Name('John', 'Smith'),
      Address('10 Merchiston', 'Edinburgh', 'EH10 5DT'),
     '21-Feb-95',
     'BEng Computing',
     'Computing'
     );
```

Subtypes-- Cont'd

- A supertype can be changed even after some subtypes have been created

alter type *peopleType*

add attribute (*gender* **varchar2(8)) cascade**;

- The **cascade** option propagates a type change to dependent types and tables

Subtypes-- Cont'd

- The supertype must be **not final**
- If it is **final**, it must be changed to **not final**

alter type *peopleType* **not final cascade**;

Constraint

A primary key can be added:

```
ALTER TABLE employee_table
  ADD (CONSTRAINT empID PRIMARY KEY (emp_ID));
```

Note:

empID – the name of the constraint

emp_ID – the name of the attribute

Summary

- Nested Relations
- Data Types
  - Built-in
  - Structured
- Structured Types
- Types & Tables
  - Supertypes & Subtypes