https://doi.org/10.5281/zenodo.4696175

International Journal of Computer Science and Information Security (IJCSIS),
Vol. 19, No. 3, March 2021

BI-TEMPORAL IMPLEMENTATION IN RELATIONAL DATABASE MANAGEMENT SYSTEMS: MS SQL SERVER

Lyn Kurian[1*], Jyotsna A[2]

[1] Rajagiri School of Engineering and
Technology (Autonomous),
Kakkanad, Ernakulam, India
Lynkurian08@gmail.com, m190311@rajagiri.edu.in

[2] Rajagiri School of Engineering and
Technology (Autonomous),
Kakkanad, Ernakulam, India
jyotsnaa@rajagiritech.edu.in

***Abstract*. Traditional database management systems (DBMS) are the computation storage and reservoir of large amounts of information. The data accumulated by these database systems is the information valid at present time, valid now. It is the data that is true at the present moment. Past data is the information that was kept in the database at an earlier time, data that is hold to be existed in the past, were valid at some point before now. Future data is the information supposed to be valid at a future time instance, data that will be true in the near future, valid at some point after now. The commercial DBMS of today used by organizations and individuals, such as MS SQL Server, Oracle, DB2, Sybase, Postgres etc., do not provide models to support and process (retrieving, modifying, inserting and removing) past and future data.**

The implementation of bi-temporal modelling in Microsoft SQL Server is important to know how relational database management system handles data the bi-temporal property. In bi-temporal database, data saved is never deleted and additional values are always appended. Therefore, the paper explores one of the way we can build bi-temporal handling of data. The paper aims to build the core concepts of bi-temporal data storage and querying techniques used in bi-temporal relational DBMS i.e., from data structures to normalized storage, and to extraction or slicing of data.

The unlimited growth of data results relational data to become complicated in terms of management and storage of data. Thus, the developers working in various commercial and industrial applications should know how bi-temporal concepts apply

to relational databases, especially due to their increased flexibility in the bi-temporal storage as well as in analyzing data. Thereby, the paper demonstrates how bi-temporal data structures and their operations are applied in Relational Database Management System.

Keywords: Bi-temporal Databases, Databases, Normalization, MS SQL Server Bi-temporal Database.

1 INTRODUCTION

The most data in systems, nowadays are temporal. The trivial task is described as tracking the validity of an entity in a single point in time. But, bi-temporality includes another dimension to the system. Bi-temporal model supports two different timelines - valid time and transaction time. First one is an object's factual time and transaction time denotes when objects are known to the database. The notion is 'what you know' and 'when you knew it'.

Bi-temporality is a solution in database implementation of financial institutions, trading, legal practices or even medicine. When there is an option to return in time to an earlier period of history, some of common complexities become of little worth. For example, in medicine, bi-temporality can present more precise patient history. You can then correlate illnesses with pre-existing conditions. Furthermore law practices can significantly benefit from this. Knowing which laws where active during given time interval, it can be applied to legal artefacts.

Nevertheless, financial institutions can use bi-temporal system to comply with financial regulations - for example, having transparent trading – i.e., uniformed, consistent data and rewinding trading metrics. The first problem that can arise in a traditional system is scalability issue. Conventional database management systems approach involves adding more physical or virtual resources to the underlying server hosting the database – more CPU, more memory or more storage i.e, designed to be vertically scaled. Such architecture creates both technological and business barriers that will not be easy to solve. Second problem is adjusting to modern data structures, which in itself could be a tedious task. Variety of different services and features which not all relational databases are equipped to handle, which are leading to diverse data structures,. In addition to this, there is one more issue, that is equally important - speed.

Data saving or retrieving speed is significant to any database management system. The extension of relational database management system can be an efficient solution to this when considering the scalability and performance metrics such as speed in execution and extraction of data in old database systems.

The paper is organized as, proposed algorithm explained in section II, model in section III, bi-temporal data definition language are presented in section IV, bi-temporal data manipulation language in section V and Concluding remarks are given in section VI.

2 LITREATURE REVIEW

In the thesis, "A Generalisation Approach to Temporal Data Models and their Implementations" explains about A Temporal Relational DBMS: TimeDB. In this dissertation [2], the author showcases that, a temporal data model has:

1. Valid time and/or transaction time as well as other time lines.
2. Temporal data structures do not make any hypothesis about a specific type system or the presence of specific time attributes.
3. Temporal data structures has time stamping of all constructs supported by the model.
4. Temporal data structures can also overcome the vertical temporal anomaly.
5. A temporally complete query language or algebra which has snapshot reducible semantics.
6. Temporal constraints with snapshot reducible semantics.

The bi-temporal relational DBMS, Time DB agrees the language ATSQL2. It is the result of process of uniting three different approaches, namely

1. ATSQL2, a temporal query language on SQL.
2. Chrono Log, which introduces the concept of temporal completeness.
3. Bi-temporal ChronoSQL, featuring a bi-temporal query language.

The translation algorithm used in Time DB is, a temporal query is converted into a temporal algebra expression using the temporal set operations union ($\cup$ t), intersect ($\cap$t) and difference (\t). Each argument to set operations is either a simple algebra expression using a temporal select (σt), project (πt) and cross product ($\times$t) operation or the result of another temporal set operation. Each simple algebra expression using only a combination of a select, project and cross product operation can then be evaluated separately using a standard SELECT-FROM-WHERE statement. These intermediate results are stored in temporary tables and then used to calculate other parts of the expression.

In Time DB, a valid-time interval $I = [vts_#$ - vte_#$), closed at the lower bound and open at the upper, is mapped internally to two attributes vts_#$ (valid-time start) and vte_#$ (valid time end). Thus, each valid-time relation can have two additional (hidden) attributes. Transaction-time tables are extended accordingly. Bitemporal tables contain attributes for both valid time and transaction time. Bi-temporal Algebra Operations Time DB handles bi-temporal queries. For these queries, special operations for set union, set difference, set intersection and cross product need to be implemented. A bi-temporal query is translated to standard SQL the same way as a unitemporal query. It uses bi-temporal algebra operations instead.

In the paper [22] Modeling and Querying Multidimensional Bitemporal Data Warehouses, by Canan Eren Atay1, Gözde Alp2* states that a Bitemporal atom is defined as a triplet, where the valid and transaction time components can be deduced as a time point, a time interval, or a temporal element. A Bitemporal atom in the form of represents Valid time lower bound (VT_LB), Valid time upper bound (VT_UB) ,Transaction time lower bound (TT_LB), Transaction time upper bound (TT_UB) and Data value, respectively.

Richard T. Snodgrass, Michael H. Böhlen, Christian S. Jensen, and Andreas Steiner states in their book that many applications need to keep track of the past states of the database, often for auditing requirements. Changes are not permitted on the past states; that would disrupt secure auditing. Instead, compensating transactions are used to correct errors. When an error is encountered, often the analyst will look at the state of the database at a previous point in time to determine where and how the error occurred. The integrity of the previous states of the database is preserved, so the solution is to have the DBMS maintain transaction time automatically.

In the paper [14] Exploiting bitemporal schema versions for managing an historical medical data warehouse: A case study, by Maria-Trinidad SERNA-ENCINAS∗ Michel ADIBA proposes that for the management, storage and visualization of a data warehouse with current and historical data, in a medical environment, we propose to use bitemporal schema versions adapted to warehouse. Nevertheless, versions management is a complex task and presents several problems. The main problem for us is the versions explosion. Thus, we must establish a mechanism to control this problem. Our proposition considers:

- An operator called SetVersion that allows to apply a set of changes on a version to generate a new version.
- A set of primitives for managing schema versions and cubes, dimensions and hierarchies.
- An evolution manager responsible for different historical schema versions management.
- A set of integrity rules that restraint the primitive execution.

3 PROPOSED MODEL

The main goal presented here, is to create comprehensive SQL bi-temporal database. For achieving the goal, these criteria must be met:

- Build bi-temporal data structure.
- Investigate how to construct normalization for data and referential integrity.
- Analyze bi-temporal manipulation techniques. (insertion, change, deletion)
- Inspect bi-temporal data extraction techniques. (coalescing, slicing)
- Outline assorted bi-temporal data extraction operations and implement them.

The solution is to have the Database Management System maintain transaction time automatically, so that the integrity of the previous states of the database is preserved. To impart the bi-temporal solution in MS SQL SERVER, these are the following functions to be done:

- Creation of the Bi-temporal TABLE
- Normalization
- Insertion
- Updating
- Deletion
- Vertical and horizontal slicing (Selection and Projection) of bi-temporal data.

3.1 MODEL

EXTENDING DATABASE

The bi-temporal database modelling is an extension of temporal database which is already inbuilt in MS SQL Server 2016 onwards. The extending database should be predominant in bi-temporal nature. All the constraints and rules specified for temporal database can be incorporated into bi-temporal database. It’s crucial to be known by the developers that focusing on bi-temporality helps in much deeper analysis for data interpretation and business requirements than the conventional relational database.

Here, in the data model, bi-temporal interactions and statements are explained. The essential difference comes down to storing the two time dimensions. [2] By extending non bi-temporal or temporal database it is possible to achieve bi-temporality in MS SQL server. Due to fact that most the database functionality can be reused, you only need to ensure data integrity and provide proper query language.

4 Bi-temporal Data Definition Language

4.1 BI-TEMPORAL TABLE CREATION

Due to the temporal upward-compatibility, temporal SQL queries returns the same results when executed on bi-temporal tables as if they were executed on non-temporal tables. The selection of valid data at time instant now in the tables referenced by the query and leaving away any timestamp attributes prior to further evaluation is firstly done. A bi-temporal table supports the tracking of a temporal table along with the time-specific data storage which is of an application specific temporal table. Use bi-temporal tables to stay user-based period information also as system-based historical information. Bi-temporal tables combines as system-period temporal tables and application-period temporal tables. All the restrictions that apply to system-period temporal tables and application temporal tables also apply to bi-temporal tables.

The constraints induced on instituting data into a database, altering data in a database, or removing data from a database, is such that the transformations should reflect what is happening to data that should be happened throughout the entire lifecycle. Thus, the syntax of those transformations maintain the semantics of the data. The constraints which every valid database state must develop to these such as entity integrity, referential integrity, domain integrity, and application-specific business rules, so that the data in the database is accurately and intelligently explain what it is about to those who manipulate the database.

Statement

CREATE TABLE table name
(…. **NOT NULL PRIMARY KEY CLUSTERED**,
// data
….Foreign ID type **REFERENCES** Parent table name,
[RowEffective StartDate] **[date time] NULL**,
[RowEffectiveEndDate] **[datetime] NULL**,
[RowAssertionStartDate] **[datetime**]
GENERATED ALWAYS AS ROW START,
[RowAssertionEndDate] **[datetime]**
GENERATED ALWAYS AS ROW END,
PERIOD FOR SYSTEM TIME
([RowAssertionStartDate], [RowAssertionEndDate]))
WITH (SYSTEM VERSIONING = ON
(HISTORY TABLE = dbo.tablenameHistory)) ; [12]

Normalization

Bi-temporal approach requires a modern design modelling, than the traditional design called denormalization. Bi-temporal data has its own flaws in terms of complexity. The nature of bi-temporal data causes issues that is specific to the modelling. Typical issues are redundancy, circumlocution, contradiction, homogeneity. As the answer, [1] 6NF allows attribute values to change independently of each other, thus avoiding redundancy and other anomalies. Independence can be maintained by incorporating irreducible components. When dealing with bi-temporal data, *6NF* is a wise choice but complex in the application point of view. Or otherwise,

Time normalization

Time is distinct, and is calculated by a clock. At a certain point in time, two clocks can't be the same. For the time to be calculated in a database, a temporal database is required, which stores a collection of time-related data. Such a system eases storing, retrieving and updating historical and future data (generically referred as temporal data). Temporal data is data that changes over time. Valid time and transaction time can be supported in together and such a database is called a bi-temporal database, comparing to a non-temporal database. The temporal approach initiates additional complexities, such as schema of the model design and also the keys. It is useless if it cannot identify an application that does not evolve over time. This means that time-varying data is necessary. Built-in time management greatly increases the functionality of a database application. The database system thus maintains past, present, and future versions of data. A conventional database, without temporal property, has only the most recent data as storage. The old values are either replaced or deleted. The progress of real-world phenomena over time cannot be stored in the database as well as accessing past data is never considered as an option. Time values are linked with attributes that indicate their periods of validity. Precisely, the temporal theory usually include two types of time: valid and transaction time. Valid time is the time period during which a fact is true with reference to reality. Transaction time is the time when a fact is recorded in the database. The concept used in time normalization is the requirement that tuples are semantically independent of one another [3].

A relation is in time normal form (TNF) if and only if it is in Boyce-Codd-Normal Form and there exists no temporal dependencies among its non-key attributes. [3]

i.e., the attributes should be non-synchronous in time. If they are synchronous, decompose the table to produce the bi-temporality to each of the attribute.

5 Bi-temporal Data Manipulation Language

5.1 INSERTING DATA INTO A BI-TEMPORAL TABLE

It is same as the statement for inserting data into a temporal table.

Statement

INSERT INTO TABLE tablename
VALUES (….);

//Avoid the columns of system generated transaction time values which will be automatically generated.

5.2 UPDATING DATA INTO A TABLE

Updating data in a bi-temporal table updates rows in history table and added to its current table.

Statement

UPDATE table name
SET column name=' value '
where key=' ';

The updating is in such a way that the data once inputted is never gone from the database and the data is coalesced and added to the history table along with current table.

5.3 DELETING DATA FROM A BI-TEMPORAL TABLE

Deleting data from a bi-temporal table deletes rows from the current table, and rows are added to its associated history table and which potentially result in new rows, inserted to the bi-temporal table.

Statement

DELETE from table **WHERE**...;

5.4 QUERYING BI-TEMPORAL DATA

Vertical and Horizontal Slicing i.e., querying a bi-temporal table can return results for a specified time period. These results consists current, historic, and future data.

A new clause FOR SYSTEM TIME in the SELECT ... FROM statement, which has five new sub clauses [10] to query bi-temporal table data:

- AS OF datetime
- ALL
- FROM start datetime TO end datetime
- BETWEEN start datetime AND end datetime
- CONTAINED IN (start datetime, end datetime)

SYSTEM TIME AS OF

The AS OF sub-clause returns rows from the bi-temporal and history table that are valid up to the time specified. It gives the complete snapshot of the current values until the specified time.

SYSTEM TIME ALL

ALL returns all the rows from the current and history table.

FROM and TO clause

The sub-clause FROM-TO returns results by combing the bi-temporal and history table because this clause excludes the upper boundary of the end time.

BETWEEN AND pair

The sub-clause BETWEEN-AND returns combined results from both the bi-temporal and history tables. This is because this sub-clause includes the upper boundary of the end time.

CONTAINED IN

This sub clause gives all rows from the history table included in the range specified. This is due to the command in the statement CONTAINED IN that returns data from the history table.

Statement

SELECT ∗ FROM table name **FOR SYSTEM TIME ALL**;

// vertical slice

SELECT* FROM table name **WHERE**

//horizontal slice

SELECT * **FROM** table name
FOR SYSTEM TIME AS OF // specifying a date time − state of data
FROM TO // range − a data audit
BETWEEN AND // range − a data audit
CONTAINED IN (,) // range − a data audit
ALL // whole data

// vertical slice and horizontal slice i.e., bi-temporal slice

SELECT FROM table name
FOR SYSTEM TIME AS OF // specifying a date time − state of data
FROM TO // range − a data audit
BETWEEN AND // range − a data audit
CONTAINED IN (,) // range − a data audit
ALL // whole data
WHERE …,

6 CONCLUSION

The data in engineering is so vast that the developers should be capable of handling data maturely. Enhanced versions of viewing and manipulating data is the need of the era. Just holding a data is meant to be of no use. Data that is meaningfully sorted is the core of an application. Time is the factor which produces meaning to the data. Thus, when we can add couple of entitlements to a single data automatically motivates enhancements in the server dimensionality.

Past historical data can always be a backbone of a business which is the complete decision maker and the game changer. Temporality in terms of data in the present time narrows the view of data. Enormous amount of data seen with past, present and future adds vital role to the business. The technologies used in medicine, law, education, etc., when shown with a meaning data brings a notion on the fact and predict the nature of the fact. There comes the importance of bi-temporal implementation of data. The structure, the normalization, the upward language compatibility is an essential backup solutions to the new implementation. In spite of that, the Bi-temporal implementation can be extended to more database types - relational, key-value stores, column value stores and document-oriented systems and graph systems. Bi-temporal operations, such as insertion, change, deletion and slicing can be implemented on multiple database management systems.

All core aspects of storing bi-temporal data in MS SQL Server is implemented and as more than a model it can be a great use case in analyzing data and report generation of business applications. This research tried to cover the bi-temporal DDL and DML operations in MS-

SQL Server.

7 REFERENCES

[1] Darko Golec, Viljan Mahnič, and Tatjana Kovač, 2017. RELATIONAL MODEL OF TEMPORAL DATA BASED ON 6TH NORMAL FORM. ISSN 1330-3651 (Print), ISSN 1848-6339(Online), Tehnički vjesnik 24, 5, 1479-1489

[2] ANDREAS STEINER, "A Generalisation Approach to Temporal Data Models and their Implementations," A dissertation submitted to the SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH for the degree of Doctor of Technical Sciences, 1998.

[3] Tobias Schlaginhaufen, "Design and implementation of a database client application for inserting, modifying, presentation and export of bitemporal personal data," Diploma Thesis, April 2007

[4] Richard T. Snodgrass, 2000. *Developing Time-Oriented Database Applications in SQL*, Morgan Kaufmann Publishers San Francisco, California.

[5] Tom Johnston, 2014 "Bitemporal Data Theory and Practice," Morgan Kaufmann publications

[6] Richard T. Snodgrass , "Managing Temporal Data A Five-Part Series," A TIMECENTER Technical Report, September 3, 1998

[7] The Code Project Open License (CPOL) (https://www.codeproject.com/Articles/17637/Bitemporal-Database-Table-Design-The-Basics,)

[8] A Primer on Managing Data Bitemporally (https://www.red-gate.com/simple-talk/sql/t-sql-programming/a-primer-on-managing-data-bitemporally/)

[9] IBM (https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.admin.dbobj.doc/doc/t0058946.html)

[10] AmeenaLalani (https://www.mssqltips.com/sqlservertip/5436/options-to-retrieve-sql-server-temporal-table-and-history-data/)

[11] N.Leviss (https://bitemporal.org/2018/08/07/1-overview-of-ms-sql-bitemporal-support/)

[12] Lyn Kurian, Jyotsna A, 2020 Management of Bi-Temporal Properties of Sql/Nosql Based Architectures – A Review *International Journal of Innovations in Engineering and Technology (IJIET), Volume 17 Issue 2, pp.61-67*

[13] Mohammed Eshtay, Assam Sleit, Monther Aldwairi"Implementing Bi-Temporal Properties Into Various Nosql Database Categories", International Journal of Computing, 18(1) 2019, 45-52

[14] Maria-Trinidad SERNA-ENCINAS Michel ADIBA "Exploiting bitemporal schema versions for

managing an historical medical data warehouse: A case study", Proceedings of the Sixth Mexican International Conference on Computer Science (ENC'05) 0- 7695-2454-0/05 $ 20.00 © 2005 IEEE

[15] ”Temporal Data in Relational Database Systems: A Comparison.” https://www.researchgate.net/publication/308900130, Conference Paper, March 2016.

[16] Dabbal Singh Mahara, https://www.slideshare.net/DabalMahara1/temporal-databases97200226,Temporal databases

[17] Ameena Lalani, https://www.mssqltips.com/sqlservertip/5436/options-to-retrievesql-server-temporal-table-and-history-data/,Options to Retrieve SQL Server Temporal Table and History Data

[18] Mike Brody, 2018, https://www.dataversity.net/implementing-bitemporal-modeling best-value/,Implementing Bitemporal Modeling for the Best Value

[19] Przemek Tomczak, https://kx.com/blog/kx-insights-powering-business- decisions bitemporal-data/,Kx Insights: Powering Business Decisions with Bitemporal Data, 29 Jun 2017 — Biotemporal Data, business intelligence, Data Analytics, digitization, Time Series

[20] https://bitemporal.org/2018/08/07/1-overview-of-ms-sql-bitemporal-support/,Overview of MS SQL bitemporal support (Part 1)

[21] Tansel et al., 1993] Tansel, A. U., Clifford, J., Gadia, S., Jajodia, S., Segev, A., and Snodgrass, R. (1993). Temporal databases: theory, design, and implementation. Benjamin-Cummings Publishing Co., Inc.

[22] Canan Eren Atay, G¨ozde Alp ”Modeling and Querying Multidimensional Bitemporal Data Warehouses”, International Journal of Computer and Communication Engineering. doi: 10.17706/ijcce.2016.5.2.110-119

[23] Dmitrij Biriukov ”Implementation Aspects of Bitemporal Databases” , Master Thesis (2018)

IJCSIS Call For Papers 2021-2022

https://sites.google.com/site/ijcsis/

The topics suggested by the journal can be discussed in term of concepts, state of the art, research, standards, implementations, running experiments, applications, and industrial case studies. Authors are invited to submit complete ***unpublished papers***, which are ***not under review in any other conference or journal*** in the following, but not limited to, topic areas. All tracks are open to both research and industry contributions.

Ad Hoc & Sensor Network	**Knowledge based systems**
Ad hoc networks for pervasive communications	**Knowledge management**
Adaptive, autonomic and context-aware computing	**Location Based Services**
Advanced Computing Architectures and New	**Management information systems**
Programming Models	**Medical imaging**
Agent-based middleware	**Micro/nano technology**
Autonomic and self-managing middleware	**Middleware Issues**
B2B and B2C management	**Middleware services and agent technologies**
BioInformatics	**Mobile and Wireless Networks**
Bio-Medicine	**Mobile Computing and Applications**
Biotechnology	**Mobile networks and services**
Broadband and intelligent networks	**Multimedia Communications**
Broadband wireless technologies	**Multimodal sensing and context for pervasive**
Cloud Computing and Applications	**applications**
Collaborative applications	**Multisensor fusion**
Communication architectures for pervasive computing	**Natural Language Processing**
Communication systems	**Network management and services**
Computational intelligence	**Network Modeling and Simulation**
Computer and microprocessor-based control	**Network Performance; Protocols; Sensors**
Computer Architecture and Embedded Systems	**Networking theory and technologies**
Computer Business	**Neural Networks**
Computer Vision	**Neuro-Fuzzy and applications**
Computer-based information systems in health care	**Open Models and Architectures**
Computing Ethics	**Open Source Tools**
Context-awareness and middleware	**Operations research**
Cross-layer design and Physical layer based issue	**Optical Networks**
Cryptography	**Pattern Recognition**
Data Base Management	**Peer to Peer and Overlay Networks**
Data Mining	**Perception and semantic interpretation**
Data Retrieval	**Pervasive Computing**
Decision making	**Performance optimization**
Digital Economy and Digital Divide	**Positioning and tracking technologies**
Digital signal processing theory	**Programming paradigms for pervasive**
Distributed Sensor Networks	**systems**
E-Business	**Quality of Service and Quality of Experience**
E-Commerce	**Real-time computer control**
E-Government	**Real-time information systems**
Emerging signal processing areas	**Real-time multimedia signal processing**
Enabling technologies for pervasive systems (e.g.,	**Reconfigurable, adaptable, and reflective**
wireless BAN, PAN)	**middleware approaches**
Encryption	**Remote Sensing**
Energy-efficient and green pervasive computing	**RFID and sensor network applications**
Event-based, publish/subscribe, and message-oriented	**Scalability of middleware**
middleware	**Security and risk management**
Evolutionary computing and intelligent systems	**Security middleware**

Expert approaches Fuzzy algorithms Fuzzy logics GPS and location-based applications Green Computing Grid Networking Healthcare Management Information Technology Human Computer Interaction (HCI) Image analysis and processing Image and multidimensional signal processing Image and Multimedia applications Industrial applications of neural networks Information and data security Information indexing and retrieval Information Management Information processing Information systems and applications Information Technology and their application Instrumentation electronics Intelligent Control System Intelligent sensors and actuators Internet applications and performances Internet Services and Applications Internet Technologies, Infrastructure, Services & Applications Interworking architecture and interoperability	**Security, Privacy and Trust Security Systems and Technolgies Sensor array and multi-channel processing Sensor fusion Sensors and RFID in pervasive systems Service oriented middleware Signal Control System Signal processing Smart devices and intelligent environments Smart home applications Social Networks and Online Communities Software Engineering Software engineering techniques for middleware Speech interface; Speech processing Supply Chain Management System security and security technologies Technology in Education Theoretical Computer Science Transportation information Trust, security and privacy issues in pervasive systems Ubiquitous and pervasive applications Ubiquitous Networks User interfaces and interaction models Virtual reality Vision-based applications Web Technologies Wired/Wireless Sensor Wireless technology**